Reinforcement Learning for POMDPs based on Action Values and Stochastic Optimization

Theodore J. Perkins
Department of Computer Science
University of Massachusetts Amherst
140 Governor's Drive
Amherst, MA 01003
perkins@cs.umass.edu

Abstract

We present a new, model-free reinforcement learning algorithm for learning to control partially-observable Markov decision processes. The algorithm incorporates ideas from action-value based reinforcement learning approaches, such as Q-Learning, as well as ideas from the stochastic optimization literature. Key to our approach is a new definition of action value, which makes the algorithm theoretically sound for partially-observable settings. We show that special cases of our algorithm can achieve probability one convergence to locally optimal policies in the limit, or probably approximately correct hill-climbing to a locally optimal policy in a finite number of samples.

Introduction

Many intelligent agents face sequential decision problems that are naturally and realistically formulated as partially-observable Markov decision processes (POMDPs). Often, however, the dynamics of an agent's environment and observations are unknown. And even when these dynamics are known, finding optimal solutions is usually NP-hard or harder, depending on the type of solution required (Littman 1994; Madani, Condon, & Hanks 1999). For such problems, reinforcement learning (RL) methods are an attractive option for finding approximate solutions.

The simplest RL approach is to ignore the fact that the agent's environment is partially observable. Standard RL algorithms such as Q-Learning or Sarsa(λ) can be applied by treating observations as if they were states of a Markov decision problem (MDP). Researchers were quick to point out that this approach can lead to suboptimal behavior or, worse, that the parameters adjusted by the learning algorithm can fail to converge or even diverge (Whitehead 1992; Baird 1995). Allowing the agent to condition its learning on the recent history of observations (McCallum 1995) or to use an internal memory can sometimes alleviate these difficulties. And empirically, Sarsa(λ) and Monte Carlo approaches appear quite robust to partial observability (Pendrith & Ryan 1996; Loch & Singh 1998). However, Sarsa(λ) and Monte Carlo approaches can fail to converge even on some very simple problems (Gordon 1996). As long as the agent's representation of the environment is not fully Markov, the theoretical soundness of applying such action-value based RL algorithms is questionable.

Partly as a response to this situation, learning algorithms that perform various kinds of stochastic gradient descent on a fixed error function have been developed (Williams 1992; Baird & Moore 1999; Sutton *et al.* 2000). Usually, such algorithms search through a continuous space of stochastic policies which condition action choice on the agent's immediate observation, or on the immediate observation plus the state of an internal memory. Under appropriate conditions, these algorithms converge to a policy that is at least locally optimal. The evidence to date suggests that these algorithms learn much more slowly than action-value based RL algorithms such as Sarsa(λ), though this question is still under study.

Our aim is to provide an algorithm that is similar in design and empirical performance to the better action-value based RL algorithms, but which enjoys superior theoretical properties, similar to those of the gradient-based approaches. Our algorithm is most similar to the Monte-Carlo Exploring Starts algorithm for MDPs (Sutton & Barto 1998), and thus we call it Monte-Carlo Exploring Starts for POMDPs (MCESP). In its simplest version, MCESP maintains a table of observation-action values, which are updated based on Monte-Carlo samples of the return. Key to our algorithm is a new definition of action value, which is inspired by the fixed-point analyses of Pendrith and McGarity (1998). Under this definition, action values give information about the value of policies in a local neighborhood of the current policy. MCESP can be interpreted as a theoretically-sound algorithm for performing local search through the discrete space of policies that map observations to actions.

MCESP's free parameters can be chosen to incorporate a number of existing ideas from the stochastic optimization literature. Experiments reported in the final section, however, demonstrate that the strength of MCESP lies not only in its connections to stochastic optimization. The RL-style updating of action values can produce performance that is superior to existing, standard optimization approaches, and competitive with the best reported RL results, which are due to Sarsa(λ) (Loch & Singh 1998).

Problem Formulation

We assume the agent's environment is modeled as a POMDP (see, e.g., McCallum 1995) with an arbitrary underlying MDP, but with a finite observation set O. When the environment emits observation $o \in O$, the agent chooses an action a from a finite set $A(o)$. A deterministic, reactive policy, π, is a function that maps each $o \in O$ to some $a \in A(o)$. The agent's task is to learn a good deterministic, reactive policy, based on experience gathered in the POMDP.[1]

We consider episodic tasks, in which there is a start-state distribution and one or more terminal states, and we assume that episodes terminate with probability one under any policy. Let $\tau = \{o_0, a_0, r_0, o_1, a_1, r_1, o_2, a_2, r_2, \ldots, o_T\}$ denote a trajectory in the POMDP, where o_T is an observation corresponding to a terminal state. We assume that every policy, π, generates a well-defined probability measure, $\mu(\pi)$, over the set of all possible trajectories. This probability measure also depends on the start-state distribution of the POMDP and the underlying MDP, but we suppress these in the notation, since they are constant. Let $R(\tau) = \sum_{t=0}^{T} \gamma^t r_t$ denote the discounted return in trajectory τ, where $\gamma \in [0, 1]$ is a discount factor. We define the value of π as the expected discounted return:

$$V^\pi = E_{\tau \sim \mu(\pi)} \{R(\tau)\} \ ,$$

which we assume to be well-defined for all π. We can write the policy value more briefly as $V^\pi = E^\pi\{R(\tau)\}$.

An important aspect of MCESP is its interpretation as a local search algorithm. We consider policies π and π' neighbors if they assign the same action to all observations except one. π is locally optimal if $V^\pi \geq V^{\pi'}$ for all neighbors π'. More generally, π is ϵ-locally optimal if $V^\pi + \epsilon \geq V^{\pi'}$ for all neighbors π'.

Defining Action Value

In MDPs, the value of state-action pair (s, a) with respect to policy π is usually defined as the expected discounted return if the environment starts in state s, the agent takes action a, and then follows π afterward (Sutton & Barto 1998). Our definition of observation-action values for POMDPs differs in several key respects. We present our definition first, and then contrast it with the standard definition for MDPs.

Let τ be a trajectory and o an observation. We define $R_{pre-o}(\tau)$ to be the portion of $R(\tau)$ coming before the first occurrence of o in τ, if any; and we define $R_{post-o}(\tau)$ to be the portion of $R(\tau)$ following the first occurrence of o in τ, if there is one, and zero otherwise. For example, if o first occurs in τ at time step j, then $R_{pre-o}(\tau) = \sum_{i=0}^{j-1} \gamma^i r_i$ and $R_{post-o}(\tau) = \sum_{i=j}^{\infty} \gamma^i r_i$. Note that for any o, we can rewrite the value of a policy as:

$$\begin{aligned} V^\pi &= E^\pi \{R(\tau)\} \\ &= E^\pi \{R_{pre-o}(\tau)\} + E^\pi \{R_{post-o}(\tau)\} \ . \end{aligned}$$

This motivates our definition of the value of observation-action pair (o, a) with respect to a policy π. Let $\pi \leftarrow (o, a)$ represent the policy that is identical to π except that observation o is mapped to action a. (Of course, $\pi \leftarrow (o, \pi(o))$ is just π.) Then we define:

$$Q^\pi_{o,a} = E^{\pi \leftarrow (o,a)} \{R_{post-o}(\tau)\} \ .$$

In words, $Q^\pi_{o,a}$ is the portion of the expected return that follows the first occurrence of o, if the agent takes action a whenever it observes o and adheres to π otherwise. This definition differs in three key respects from the standard state-action value definition for MDPs. First, the notion of starting in state s is replaced by a first occurrence of observation o. Second, the agent does not take action a and then follow π afterward. The agent takes action a and follows $\pi \leftarrow (o, a)$ afterward. In other words, the agent takes action a every time it sees observation o. Third, the observation-action value is the portion of the policy's expected discounted return that follows o, not the discounted return following o itself. This makes a difference in how the discount factor comes into play if o's first occurrence can happen at different times in different trajectories—the later o occurs, the less the return following o contributes to the overall policy value. Consider, for example, two trajectories, with reward sequences $\{r_0, r_1, r_2, \ldots, r_T\}$ and $\{r'_0, r'_1, r'_2, \ldots, r'_{T'}\}$. Suppose that in the first trajectory, o occurs on time step 2, and in the other, o occurs on time step 4. Then the first trajectory contributes $\gamma^2 r_2 + \gamma^3 r_3 + \ldots + \gamma^T r_T$ to the observation-action value, and the second trajectory contributes $\gamma^4 r'_4 + \gamma^5 r'_5 + \ldots + \gamma^{T'} r'_{T'}$ Under the standard definition, the trajectories would contribute $r_2 + \gamma r_3 + \ldots + \gamma^{T-2} r_T$ and $r'_4 + \gamma r'_5 + \ldots + \gamma^{T'-4} r'_{T'}$ respectively.

In MDPs, a policy is optimal if and only if it is greedy with respect to its action values (as normally defined). This is not necessarily true of the action values learned by Q-Learning or Sarsa(λ), for example, when applied to POMDPs. The theoretical motivation for our definition of action value is that it preserves this property, to some degree, in POMDPs.

Theorem 1 *For all π and $\pi' = \pi \leftarrow (o, a)$,*

$$V^\pi + \epsilon \geq V^{\pi'} \iff Q^\pi_{o,\pi(o)} + \epsilon \geq Q^\pi_{o,a} \ .$$

Proof: Let π be any policy and let $\pi' = \pi \leftarrow (o, a)$ be a neighboring policy. Then:

$$\begin{aligned} & V^\pi + \epsilon \geq V^{\pi'} \\ \iff \quad & E^\pi \{R_{pre-o}(\tau)\} + E^\pi \{R_{post-o}(\tau)\} + \epsilon \\ & \quad \geq E^{\pi'} \{R_{pre-o}(\tau)\} + E^{\pi'} \{R_{post-o}(\tau)\} \\ \iff \quad & E^\pi \{R_{post-o}(\tau)\} + \epsilon \geq E^{\pi \leftarrow (o,a)} \{R_{post-o}(\tau)\} \\ \iff \quad & Q^\pi_{o,\pi(o)} + \epsilon \geq Q^\pi_{o,a} \ . \end{aligned}$$

The middle equivalence holds because the portion of the expected discounted return before the first occurrence of o cannot depend on the action taken from observation o. □

Corollary 1 *A policy is locally optimal if and only if it is greedy with respect to its action values (as we have defined them). A policy π is ϵ-locally optimal if and only if $Q^\pi_{o,\pi(o)} + \epsilon \geq Q^\pi_{o,a}$ for all o and a.*

[1] More generally, the POMDP's observation set can be infinite and the agent's actions can depend on the history of observations, rewards, and actions—as long the agent can be described as mapping each history to one of a finite number of table entries and associating to each entry a value for each action. That is, the agent aggregates the set of all histories into a finite number of partitions.

MCESP(Q,π,α,ϵ)
Inputs: initial action values Q, policy π that is greedy w.r.t. Q, and learning rate and comparison threshold schedules α and ϵ.

$c_{o,a} \leftarrow 0$ for all o and a.
$n \leftarrow 0$
repeat
  Choose some o and $a \in A(o)$.
  Generate a trajectory, τ, according to $\pi \leftarrow (o, a)$.
  $Q_{o,a} \leftarrow (1-\alpha(n, c_{o,a}))Q_{o,a} + \alpha(n, c_{o,a})R_{post-o}(\tau)$
  $c_{o,a} \leftarrow c_{o,a} + 1$
  if $\max_{a'} Q_{o,a'} - \epsilon(n, c_{o,a'}, c_{o,\pi(o)}) > Q_{o,\pi(o)}$ **then**
    $\pi(o) \leftarrow a' \in \arg\max_{a'} Q_{o,a'} - \epsilon(n, c_{o,a'}, c_{o,\pi(o)})$
    $n \leftarrow n + 1$
    $c_{o'',a''} \leftarrow 0$ for all o'' and a''
  end if
until Termination

Figure 1: The MCESP algorithm.

The MCESP Algorithm

In this section, we present the MCESP learning algorithm, which is based on the definition of action value above. The algorithm is displayed in Figure 1. It maintains a table of action values, Q, a current policy, π, a count of the number of times the current policy has changed, n, and counts, c, of the number of times each observation-action pair has been updated since the last policy change. At the beginning of each trial, the algorithm chooses some observation-action pair (o, a) to "explore." It follows the policy $\pi \leftarrow (o, a)$ for the whole trial, producing a trajectory, τ. The action value $Q_{o,a}$ is updated based on $R_{post-o}(\tau)$, and the algorithm checks if the current policy should change. The learning rates for action value updates follow a schedule, α, which depends on the number of policy changes since the algorithm began and on the number of updates the action value has received since the policy last changed. When checking whether the current policy should change as a result of an update, the algorithm compares the on-policy action value, $Q_{o,\pi(o)}$, with the off-policy action values $Q_{o,a'}$ for $a' \neq \pi(o)$. For a change to occur, it requires that $Q_{o,a'} > Q_{o,\pi(o)} + \epsilon$, where ϵ is a comparison threshold. (In Figure 1 this is written as $Q_{o,a'} - \epsilon > Q_{o,\pi(o)}$.) The ϵ allows one to express, for example, that an off-policy action must appear significantly better than the on-policy action before a change is made. Comparison thresholds are allowed to vary, depending on the number of policy changes so far and the number of times each of the action values involved in the comparison have been updated since the last policy change.

This general presentation of the algorithm leaves open a number of choices: how observation-action pairs are chosen for exploration, how learning rates and comparison thresholds are scheduled, and under what conditions the algorithm terminates. By making different choices for these, MCESP can incorporate various ideas from reinforcement learning and stochastic optimization. We begin by discussing a version of MCESP that estimates action values by taking a fixed number of samples and then comparing the sample averages. Next, we describe a version of MCESP based on Greiner's PALO algorithm which offers a PAC-style guarantee of hill-climbing to a local optimum. And lastly, we describe a set of conditions that ensure MCESP converges to a locally optimal policy in the limit. There are many other specializations of MCESP that might be of interest, and we mention some of these in the conclusion section.

The Sample Average Approximation

If the agent does not know the dynamics of the POMDP it must solve, then it cannot exactly evaluate any policy. But it can estimate a policy's value by generating some fixed number of trajectories, k, under the policy and computing the average discounted return. In the stochastic optimization literature, this has been called the sample average approximation (e.g., Kleywegt *et al.* 2001). The idea has also appeared in a number of PAC-style results in the RL literature (e.g., Ng and Jordan 2000).

A local search procedure based on this principle proceeds in stages, where at each stage k samples are taken of the value of a current policy and of each neighboring policy. If no neighbor has a better sample average than the current policy, the algorithm terminates. Otherwise, the neighbor with the best average becomes the current policy for the next stage. MCESP can reproduce this behavior by: 1) choosing observation-action pairs for exploration in simple round-robin fashion; 2) letting $\alpha(n, i) = \frac{1}{i+1}$, corresponding to simple averaging of the sample discounted returns; 3) letting $\epsilon(n, i, j) = +\infty$ if $i < k$ or $j < k$ and 0 otherwise, which effectively rules out comparison if fewer than k samples have been taken of either action value; 4) terminating if no policy changes are recommended after taking k samples of the value of each observation-action pair. If these particular choices are made, we call the resulting algorithm MCESP-SAA.

If k is small, then one expects the action-value estimates to be poor. MCESP-SAA could easily switch to a worse policy or stop erroneously at a policy that is not locally optimal. If k is large, then action-value estimates should be good, and MCESP-SAA should move strictly uphill and stop at the first locally-optimal solution it encounters. The next version of MCESP that we consider provides a PAC-style guarantee of the latter type of behavior.

PAC Hill-Climbing

Greiner's PALO algorithm is a general method for hill-climbing in the solution space of a stochastic optimization problem with finite local neighborhood structure (Greiner 1996). Given any ϵ and δ, PALO traverses, with probability at least $1 - \delta$, a sequence of solutions that is of strictly improving quality and terminates at a solution that is ϵ-locally optimal. At each stage, n, PALO determines a number, k_n, of samples that should be taken of the value of each solution in the current neighborhood. After k_n samples, PALO applies a simple threshold test based on Hoeffding's inequality to determine if any neighbor is sufficiently better than the current solution to warrant a switch. PALO also includes more stringent tests that allow it to move to a neighbor or terminate before a full k_n samples are taken, if the evidence is overwhelming. The number of samples taken at each step, k_n, is polynomial in $\frac{1}{\epsilon}$ and logarithmic in $\frac{1}{\delta}$ and n.

MCESP can implement PALO's strategy. As in the previous section, MCESP should sample observation-action pairs in round-robin fashion, and use learning rates that correspond to simple averaging (i.e., $\alpha(n,i) = \frac{1}{i+1}$). Suppose that N is an upper bound on the size of any policy's neighborhood, and suppose that all samples of observation-action values fall in some range $[x,y] \subset \Re$. Let $\delta_n = \frac{6\delta}{n^2\pi^2}$ and $k_n = \left\lceil 2\frac{(y-x)^2}{\epsilon^2} \ln \frac{2N}{\delta_n} \right\rceil$. Then MCESP reproduces PALO's comparison tests with the threshold schedule:

$$\epsilon(n,i,j) = \begin{cases} (y-x)\sqrt{\frac{1}{2i}\ln\left(\frac{2(k_n-1)N}{\delta_n}\right)} & \text{if } i = j < k_n \\ \frac{\epsilon}{2} & \text{if } i = j = k_n \\ +\infty & \text{otherwise .} \end{cases}$$

Note that thresholds are only finite when $i = j$. In PALO's formulation, "one sample" constitutes one sample from every member of the current neighborhood, thus there is never an issue of some members having been sampled more times than others. We could, of course, allow for comparisons when $i \neq j$. But to remain faithful to PALO as originally defined, we choose not to do so. The algorithm terminates if, after k_n samples, no policy change is triggered, or if, after i samples have been taken of each action value,

$$Q_{o,a} < Q_{o,\pi(o)} + \epsilon - (y-x)\sqrt{\frac{1}{2i}\ln\left(\frac{2(k_n-1)N}{\delta_n}\right)} ,$$

for all o and all $a \neq \pi(o)$. We call this version MCESP-PALO. The theorem below follows from Theorem 1 of Greiner (1996) and Corollary 1.

Theorem 2 *For any $\delta \in (0,1)$ and $\epsilon > 0$, with probability at least $1-\delta$, MCESP-PALO traverses a sequence of policies $\pi_0, \pi_1, \ldots, \pi_T$ and terminates, where each policy is of strictly greater value than the previous policy and π_T is ϵ-locally optimal.*

Probability One Convergence to Locally Optimal Policies

Because samples of action values are stochastic, it is not possible to guarantee that a policy is ϵ-locally optimal after any finite number of samples. Suppose, however, that MCESP takes samples of a particular policy's action values indefinitely. If the policy is not ϵ-locally optimal, then eventually the action value estimates should indicate that fact, prompting a switch to another policy. If, after n policy changes, MCESP delays comparing action values until k_n samples are taken, and if $\lim_{n\to\infty} k_n = \infty$, then MCESP should eventually converge to an ϵ-locally optimal policy. As n increases, the precision of the action value estimates by the time they start to be compared increases. For high enough n, MCESP should never mistakenly leave a strictly ϵ-locally optimal policy and should always leave a non-ϵ-locally optimal policy for some better one. Interestingly, it is not necessary to wait for increasing periods before comparing action values, nor is it necessary to combine samples using simple averaging, or to use "slow" learning rates so that data has a chance to accumulate before a change can occur.

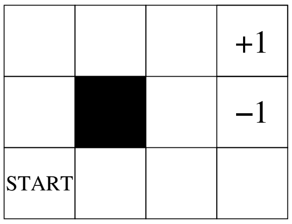

Figure 2: Parr's and Russell's Gridworld

Theorem 3 *If the initial action value estimates and all samples of action values are contained in some $[x,y] \subset \Re$; $\epsilon(n,i,j) = \epsilon_0$ for all n, i, j; $l_i \leq \alpha(n,i) \leq u_i$ for all n, where $\{l_i\}$ and $\{u_i\}$ are Robbins-Monroe sequences[2]; and observation-action pairs are selected for exploration at random with each pair having at least probability $p_{min} > 0$ of being selected on any trial; then with probability 1, MCESP converges to an ϵ_0-locally optimal policy (i.e., an ϵ_0-locally optimal policy becomes the current policy at some time, and remains the current policy forever).*

We call this version MCESP-CE, for "constant epsilon." This theorem does not follow directly from existing theory, and, unfortunately, our proof is rather detailed and does not fit the space available. A full proof will be available in a technical report.

Our theorems on MCESP-PALO and MCESP-CE discuss termination or convergence to ϵ-locally optimal policies. We note that, since the policy space is finite, for sufficiently small ϵ, all ϵ-locally optimal policies are simply locally optimal. However, that ϵ may not be known, and a more theoretically pleasing solution is to let comparison thresholds decrease over time, so that convergence to locally optimal policies is guaranteed. We are presently working on schedules for achieving this goal.

Experiments

We experimented with MCESP and Sarsa(λ) in Parr's and Russell's partially observable gridworld domain (Parr & Russell 1995). This problem is small enough that we were able to exactly evaluate every possible policy using off-line dynamic programming. This yields a deeper understanding of the problem and sheds light on the experimental results we obtained. At the same time, the problem is large enough to demonstrate interesting structure of the sort expected in larger, more realistic domains.

The domain is depicted in Figure 2. The 11 open squares represent possible locations of the agent. Each trial starts with the agent in the "START" square and ends after 100

[2] $\{x_i\}_{i=0}^{\infty}$ is a Robbins-Monroe sequence if $x_i \in [0,1]$ for all i, $\sum_i x_i = \infty$, and $\sum_i x_i^2 < \infty$.

timesteps or when the agent enters the $+1$ or -1 squares, which give terminal rewards of $+1$ and -1 respectively. All other transitions receive a reward of -0.04. In each state, the agent may take one of four actions: Up, Right, Down, or Left, which move the agent in the named direction with probability 0.8, and in one of the perpendicular directions, each with probability 0.1. If one of these transitions would take the agent outside the maze or into the black square, the agent stays in place instead. The problem is partially observable because the agent only observes whether or not there is an open square to the left and whether or not there is on open square to the right. Thus, the POMDP has 11 states, 4 non-terminal observations, and just $4^4 = 256$ reactive policies.

Using dynamic programming to compute the values of all 256 reactive policies, we found that 118 have zero probability of reaching either of the terminal states. Such policies have value -4.0. 17 of those policies are non-strict local optima, being completely surrounded by other policies of their kind; we call these the bad locally optimal policies. There is a single globally-optimal policy with value 0.303470. Thus, the policy space is qualitatively characterized as having a plateau of minimal-value policies, some of them locally optimal, and a single peak corresponding to the unique, globally-optimal policy.

We experimented with various versions of MCESP: MCESP-SAA with $k \in \{1, 2, \ldots, 9\} \cup \{10, 20, \ldots, 90\} \cup \{100, 200, \ldots, 1000\}$; MCESP-PALO with $\epsilon = 0.01$ and $\delta = 0.99$; and MCESP-CE with uniformly random exploration, $\epsilon(n, i, j) = 0.01$ and $\alpha(n, i) = bi^{-p}$ for $b \in \{0.1, 0.25, 0.5, 1.0\}$ and $p \in \{0.51, 0.6, 0.8, 1.0\}$. The δ value for MCESP-PALO is quite high, but even at this level, $k_0 = 1,986,744$. The Hoeffding bound upon which PALO's equations are based is fairly loose, and MCESP-PALO tends to take a large number of samples before achieving the confidence to make a policy change. For MCESP-CE we report results just for $(b, p) = (1, 1)$, corresponding to simple averaging, and $(b, p) = (0.1, 0.6)$, which had the best asymptotic performance. We also ran Sarsa(λ) in the same manner as described in Loch and Singh (Loch & Singh 1998); We refer the reader to their paper for details. Loch's and Singh's results with Sarsa(0.9) are the best reported for model-free RL on a variety of problems, including the Parr and Russell gridworld. They found that Sarsa quickly and consistently found the optimal policy, so this is a high standard to compare to.

Figure 3 presents the result of our experiments. For MCESP-SAA and MCESP-PALO, which are terminating algorithms, we plot the mean value of the final policy against the time to termination, measured in total timesteps, or actions taken, in the POMDP. The set of white boxes corresponds to MCESP-SAA, with k increasing from left to right. For small k, the algorithm often erroneously terminates at policies that are not locally optimal, leading to poor average performance. For higher k, MCESP-SAA rarely moves to a lower-value policy, but often gets trapped in the set of bad locally-optimal policies. MCESP-PALO behaves essentially like MCESP-SAA with very large k. In the 1000 runs, we never observed MCESP-PALO making a change leading to a worse policy, although in theory there is some chance of

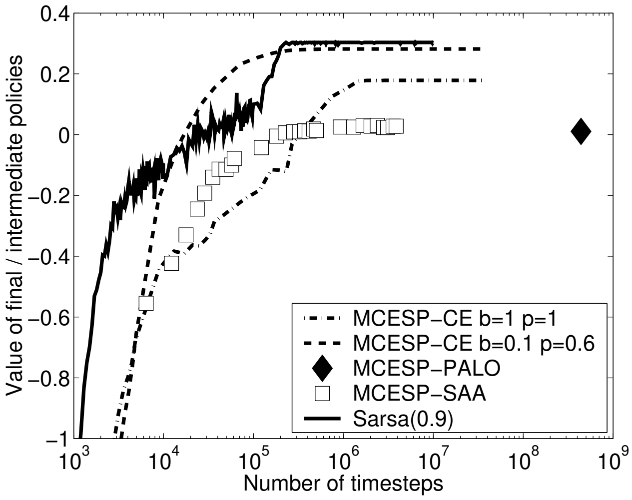

Figure 3: Results on Parr's and Russell's Gridworld

this happening. 932 runs ended at the optimal policy, with 68 ending at one of the bad locally-optimal policies.

For MCESP-CE and Sarsa we plot the mean current-policy value over the course of the runs. Every run of Sarsa converged to the optimal policy by around the $200,000^{th}$ step (about 10,000 trials). Encouragingly, MCESP-CE with $(b, p) = (0.1, 0.6)$ performed comparably. Asymptotic performance was slightly worse, as 5 of 1000 runs converged to one of the bad locally-optimal policies. Learning speed was comparable, which was surprising since MCESP updates only one action value per trial, which comes out to about once per twenty timesteps. These results are also encouraging because we spent little time hand-tuning the parameters of MCESP. By contrast, we spent approximately two days trying to get Sarsa(λ) to behave well before we gave up and implemented the exact design of Loch and Singh (1998). It turns out that Sarsa's behavior is very sensitive to how actions are chosen for exploration. Not knowing this, we spent a lot of time trying different choices for λ, different learning rates, and different versions of accumulating and replacing eligibility traces (Sutton & Barto 1998).

MCESP-CE with $(b, p) = (1, 1)$ did not fare as well as Sarsa(λ) or the other version of MCESP-CE, though performance was decent. 29 runs converged to one of the bad policies, with the rest converging to the optimal policy. Comparing the two versions of MCESP-CE it appears that using learning rate schedules other than simple averaging can be beneficial. Using simple averaging, the first update to an action value after a policy change completely wipes out the old action value. By contrast, the $(b, p) = (0.1, 0.6)$ schedule allows action value information to persist across policy changes—the first update, for example, is averaged in to the old action value with a learning rate of 0.1. This can be a good idea if, for example, changing the action associated to one observation is unlikely to affect the relative (or absolute) values of other observation-action values. Such a schedule might also be expected to reduce variance, which is often an issue with Monte Carlo algorithms.

Conclusion

We formulated a new definition of action value for reinforcement learning of reactive policies for POMDPs, and presented a general algorithm, MCESP, based on this definition. MCESP is similar in design to action-value based RL algorithms such as Q-Learning, Sarsa(λ), and especially Monte-Carlo Exploring-Starts (Sutton & Barto 1998). But, unlike those algorithms, MCESP is provably sound for application to partially observable settings. We have shown that MCESP can be specialized to achieve PAC hill-climbing to locally optimal policies, or probability one convergence in the limit to locally optimally policies.

Such theoretical guarantees are a double-edged sword. In experiments, we found that all versions of MCESP suffered to some degree from converging to locally-but-not-globally optimal policies. Encouragingly, though, one version of MCESP performed comparably in terms of learning speed and mean solution quality to Sarsa(λ), which was previously reported to be the best-performing model-free RL algorithm for learning reactive policies for POMDPs (Loch & Singh 1998). Further experimental work is needed to see how well this result generalizes, particularly for more complex domains than the one studied.

Our use of MCESP to date has been unrefined in many ways. For example, in all cases we have used simple round-robin or uniformly-random exploration. But there is every reason to believe that more sophisticated exploration strategies could be beneficial in identifying superior policies more quickly (Kaelbling 1993; Maron & Moore 1994). We are also interested in the possibility of letting comparison thresholds be negative at times, allowing occasional moves to policies that may seem worse. This idea has proven very successful in the simulated annealing algorithm for combinatorial optimization, and could be useful in RL as well.

Acknowledgments

This work was supported in part by the National Science Foundation under Grant Nos. ECS-0070102 and ECS-9980062. We thank Daniel Bernstein and Doina Precup for comments on drafts of this document.

References

Baird, L. C., and Moore, A. W. 1999. Gradient descent for general reinforcement learning. In *Advances in Neural Information Processing Systems 11*. MIT Press.

Baird, L. C. 1995. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*, 30–37. Morgan Kaufmann.

Gordon, G. 1996. Chattering in Sarsa(λ). CMU Learning Lab Internal Report. Available at www.cs.cmu.edu/$\sim$ggordon.

Greiner, R. 1996. PALO: A probabilistic hill-climbing algorithm. *Artificial Intelligence* 84(1–2):177–204.

Kaelbling, L. P. 1993. *Learning in embedded systems*. Cambridge, MA: MIT Press.

Kleywegt, A. J.; Shapiro, A.; and de Mello, T. H. 2001. The sample average approximation method for stochastic discrete optimization. *SIAM Journal of Optimization* 12:479–502.

Littman, M. L. 1994. Memoryless policies: Theoretical limitations and practical results. In *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*. Cambridge, MA: MIT Press.

Loch, J., and Singh, S. 1998. Using eligibility traces to find the best memoryless policy in a partially observable Markov decision process. In *Proceedings of the Fifteenth International Conference on Machine Learning*. San Francisco, CA: Morgan Kaufmann.

Madani, O.; Condon, A.; and Hanks, S. 1999. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision process problems. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*. Cambridge, MA: MIT Press.

Maron, O., and Moore, A. 1994. Hoeffding races: Accelerating model selection search for classification and function approximation. In *Advances in Neural Information Processing Systems 6*, 59–66.

McCallum, A. K. 1995. *Reinforcement learning with selective perception and hidden state*. Ph.D. Dissertation, University of Rochester.

Ng, A., and Jordan, M. 2000. PEGASUS: A policy search method for large MDPs and POMDPs. In *Uncertainty in Artificial Intelligence, Proceedings of the Sixteenth Conference*.

Parr, R., and Russell, S. 1995. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*. San Francisco, CA: Morgan Kaufmann.

Pendrith, M. D., and McGarity, M. J. 1998. An analysis of direct reinforcement learning in non-Markovian domains. In *Machine Learning: Proceedings of the 15th International Conference*, 421–429.

Pendrith, M. D., and Ryan, M. R. K. 1996. Actual return reinforcement learning versus temporal differences: Some theoretical and experimental results. In Saitta, L., ed., *Machine Learning: Proceedings of the 13th International Conference*, 373–381.

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

Sutton, R. S.; McAllister, D.; Singh, S.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*. MIT Press.

Whitehead, S. D. 1992. *Reinforcement Learning for the Adaptive Control of Perception and Action*. Ph.D. Dissertation, University of Rochester.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8:229–256.