SarsaLandmark: An Algorithm for Learning in POMDPs with Landmarks

Michael R. James
AI and Robotics, Technical Research Dept.
Toyota Research Institute NA
Ann Arbor, MI, USA
michael.r.james@gmail.com

Satinder Singh
Computer Science and Engineering
University of Michigan
Ann Arbor, MI, USA
baveja@umich.edu

ABSTRACT

Reinforcement learning algorithms that use eligibility traces, such as Sarsa(λ), have been empirically shown to be effective in learning good estimated-state-based policies in partially observable Markov decision processes (POMDPs). Nevertheless, one can construct counterexamples, problems in which Sarsa($\lambda < 1$) fails to find a good policy even though one exists. Despite this, these algorithms remain of great interest because alternative approaches to learning in POMDPs based on approximating belief-states do not scale. In this paper we present SarsaLandmark, an algorithm for learning in POMDPs with "landmark" states (most man-made and many natural environments have landmarks). SarsaLandmark simultaneously preserves the advantages offered by eligibility traces and fixes the cause of the failure of Sarsa(λ) on the motivating counterexamples. We present a theoretical analysis of SarsaLandmark for the policy evaluation problem and present empirical results on a few learning control problems.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Theory

Keywords

reinforcement learning, partial observability, POMDP, landmark

1. INTRODUCTION

Sequential decision problems in which the agent can sense the complete state of the environment can be modeled as Markov decision processes (MDPs) for which considerable theoretical and empirical progress has been achieved in developing planning and reinforcement learning algorithms [12]. The picture is far less rosy for problems in which the agent cannot sense the complete state of the environment. Such agent-environment interactions suffer from what is called perceptual aliasing (or hidden state) and are modeled as *partially observable* MDPs or POMDPs [4]. Most real-world sequential decision problems are POMDPs and thus solving them efficiently is of great interest and significance to AI.

A central idea behind most approaches to planning and learning in POMDPs is to treat the problem as an MDP whose state is the agent's state of information, or belief-state. While there have been advances in approximate algorithms for these problems, there is still a fundamental issue in scaling these approaches because a POMDP with n underlying discrete states converts to a much larger MDP that has a $(n-1)$-dimensional continuous state space.

Alternative, more successful, approaches to learning in POMDPs estimate (usually discrete) state from the history of interaction and then use the temporal difference (TD) family of algorithms (Sutton, 1988) to learn policies conditioned on the estimated states. In particular, TD algorithms that use eligibility traces, e.g., Sarsa(λ), have been empirically shown to be effective at finding good estimated-state-based policies in POMDPs [6, 8, 9]. Nevertheless, one can construct simple counterexample POMDPs in which Sarsa(λ) fails to find good policies even though they exist. Despite this, these algorithms remain of great interest because of the scaling difficulties with belief-state approaches.

In this paper, we present SarsaLandmark, an algorithm for learning in POMDPs with "landmarks" or special states that are not hidden to the agent. Most man-made environments are engineered to have landmarks, e.g., street names at road intersections, labels at doorways in offices and at entrances to buildings, RF beacons in robotic environments, etc. Even in purely natural environments, we often look for landmarks by looking for perceptually salient and distinct states. Our algorithm, SarsaLandmark, exploits the presence of these landmarks to overcome a cause of the failure of Sarsa(λ) on the motivating counterexample while preserving the intuitive advantages of eligibility traces in hidden state environments.

Our main contributions in this paper are *a)* an example that provides an explanation of how Sarsa(λ) can fail in POMDPs (in Section 4), *b)* SarsaLandmark, an algorithm that adapts Sarsa(1) to landmark-POMDPs (in Section 5), *c)* a theoretical analysis of SarsaLandmark for the policy evaluation problem (in Section 6), and *d)* empirical results for SarsaLandmark on three learning control problems (in Section 7). These contributions are significant because landmark-POMDPs are fairly ubiquitous and because SarsaLandmark should scale as well as Sarsa(λ) and perform

Cite as: SarsaLandmark: An Algorithm for Learning in POMDPs with Landmarks, Michael R. James, Satinder Singh, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 585–592

better in such problems.

2. POMDP BACKGROUND

In this section we briefly describe the POMDP formulation of reinforcement learning problems. A POMDP is specified by a tuple $< S, A, O, P, B, R, \gamma >$. The environment can be in one of a finite set of states S, the agent can choose from among a finite set of actions A, and the agent's sensors provide it with an observation from a finite set O. On executing action $a \in A$ in state $s \in S$ the agent receives expected payoff $R^a(s)$ and the environment transitions to a random next state $s' \in S$ with probability $P(s'|s,a)$. The probability of the agent observing $o \in O$ given that the hidden state of the environment is $s \in S$ is denoted by $B(o|s)$. The agent's goal is to choose actions in such a way as to maximize the expected value of the discounted sum of payoffs over an infinite horizon, i.e., maximize

$$E\{\sum_{t=0}^{\infty} \gamma^t r_t\}$$

where r_t is the payoff received at time step t, and $0 \leq \gamma < 1$ is a discount factor that makes future payoffs less valuable than the immediate payoffs.

Landmark POMDPs: In this paper, we consider Landmark-POMDPs, i.e., POMDPs with one or more landmark states.

Definition 1. *A landmark state is a state that has a unique observation: state $s \in S$ is a landmark state, if $\exists\, o \in O$ such that $B(o|s) = 1$ and $\forall\, s' \neq s$, $B(o|s') = 0$.*

We will assume that the agent knows when it is in a landmark state and when it is not. Indeed, a state that has a unique observation but this fact is unknown to the agent will not be treated as a landmark state. Note that if all the states of a POMDP are landmark states then it is fully observable and is in fact an MDP.

In MDPs it is known that the agent can choose optimal actions as a function of the most recent state alone, i.e., there exists an optimal policy of the form $S \rightarrow A$. Consequently TD algorithms without eligibility traces, such as Q-learning are able to learn optimal policies from sufficiently exploratory experience with an environment [13]. However, even in MDPs, using eligibility traces can confer a computational advantage over not using them [12] in the form of faster convergence to the optimal policy.

In POMDPs, on the other hand, the optimal action may depend on the history of action-observation pairs rather than just on the most recent observation. Learning a mapping from arbitrarily long histories to actions would be infeasible. Therefore, most approaches to learning in POMDPs extract a kind of "estimated state" from the history of observations, often simply by remembering a very small number of past observations, and then learning policies that map such estimated-state to actions.

Except in the simplest of cases, the learning problem remains a POMDP even with estimated states, and it is known that algorithms that don't use eligibility traces fare badly in POMDPs [10]. Loch and Singh (1998) empirically showed that Sarsa(λ), which does use eligibility traces, is often able to overcome the presence of hidden state and learn good estimated-state-based policies (see Perkins & Precup, 2002) for further empirical and theoretical corroboration of this).

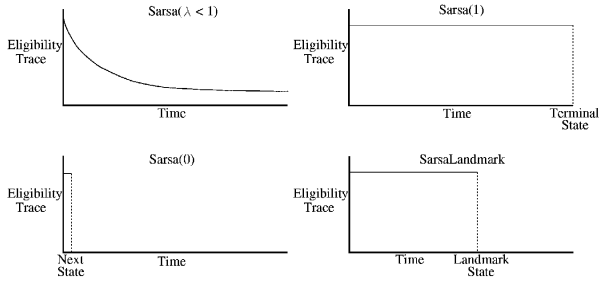

Figure 1: This figure shows the eligibility trace for four different algorithms: Sarsa($\lambda < 1$), Sarsa(0), Sarsa(1), and SarsaLandmark.

Next, we describe the Sarsa(λ) family of algorithms.

3. SARSA(λ) BACKGROUND

Sarsa(λ) maintains an eligibility trace, η, as well as a Q-value, Q, for every (estimated-state, action) pair. Hereafter, we will use x to denote estimated-states. On experiencing transition $< x_t, a_t, r_t, x_{t+1} >$ at time step t, the following updates are performed in order:

$$\begin{aligned} \eta_t(x_t, a_t) &= 1 \\ \eta_t(x, a) &= \gamma\lambda\eta_{t-1}(x, a),\ \forall\, (x \neq x_t \text{ or } a \neq a_t) \\ Q_{t+1}(x, a) &= Q_t(x, a) + \alpha\eta_t(x, a)\delta_t,\ \forall\, x, a, \end{aligned} \quad (1)$$

where $\delta_t = r_t + \gamma Q_t(x_{t+1}, a_{t+1}) - Q_t(x_t, a_t)$ is the TD-error at time step t, α is the step size, and $0 \leq \lambda \leq 1$ is a parameter giving rise to a family of algorithms. The eligibility trace value $\eta(x, a)$ determines how much credit/blame is assigned to each (x, a) for the TD-error. The Q-values can be initialized to any value. The eligibility traces are initialized to zero, and in episodic tasks they are reinitialized to zero after every episode. At every time step, the eligibility trace for each (x, a) is reduced by a multiplicative factor of λ. Thus, looking backwards in time the (x, a) that occurred more recently gets more credit for the TD-error.

In order for this algorithm to learn good policies, we have to ensure that all actions are tried often enough in each state. Often, an ϵ-greedy policy is used — such a policy takes a random action with probability ϵ and an action that is greedy with respect to the Q-values with the remaining $(1 - \epsilon)$ probability.

Note that the above formulation of Sarsa(λ) is general and applies to MDPs by letting x_t be s_t, as well as to POMDPs as long as the estimated state is discrete and finite (as one example, x_t could just be o_t).

Figure 1 shows the evolution of the eligibility trace over time for various special values of λ. For $\lambda < 1$ the eligibility of (x, a) decreases exponentially with time as in the top left-hand graph. For $\lambda = 0$ the eligibility of (x, a) lasts one time step only as shown in the bottom left-hand graph. In this case, the Q-value update at time t updates only (x_t, a_t) as follows:

$$Q_{t+1}(x_t, a_t) = Q_t(x_t, a_t) + \alpha\delta_t$$

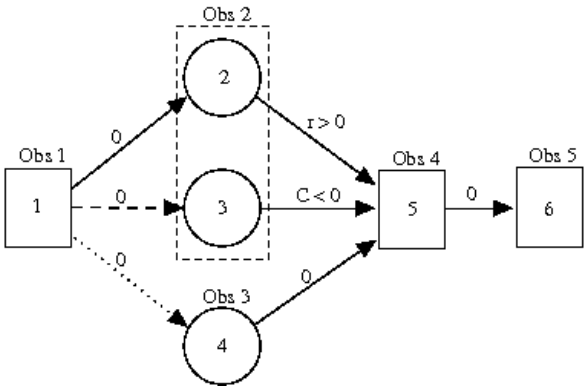

Figure 2: POMDP counterexample to Sarsa(λ). See text for details.

Reward r	$min.\lambda$
8.0	0.76
4.0	0.89
2.0	0.94
1.0	0.971
0.5	0.985
0.25	0.992
0.125	0.9962
0.0625	0.9981

Table 1: For various values of the reward r, this table shows the minimum λ in Sarsa(λ) that solves the POMDP in Figure 2 with $C = -150$.

and leaves every other $Q_t(x, a)$ unchanged. If $\lambda = 1$, we get the *Monte-Carlo* algorithm whose eligibility trace does not decay at all as seen in the top right-hand graph in Figure 1. In episodic problems the eligibility trace drops to zero when a terminal state is reached. In this case, each $Q_t(x_t, a_t)$ is updated based on the actual discounted sum of payoffs achieved from time t onwards with no role played by the Q-values of the (x, a) that appear along the trajectory.

Increasing λ from 0 to 1 interpolates between these two extremes; the smaller the λ the more the updates are influenced by the current estimate of the Q-values, and the larger the λ the greater the influence of the sequence of actual payoffs obtained. There is a bias-variance tradeoff faced in choosing a good λ: the higher the λ the higher the variance of the update, and the lower the λ the higher the "bias" introduced by the estimated Q-values. Monte-Carlo has the highest variance and no bias, while Sarsa(0) has the least variance and the most bias.

In the case of MDPs, as learning proceeds and the Q-values improve the bias introduced by using the inaccurate Q-values in the updates decreases over time. Thus, using intermediate values of λ in MDPs still allows convergence to the true Q-values. Indeed, much empirical work [12] and some theoretical work [5] has shown that intermediate values of λ provide the fastest rate of convergence. As we will see in the next section, the case for POMDPs is very different.

4. MOTIVATING COUNTEREXAMPLE

As shown in Loch & Singh (1998), Sarsa(λ) finds near-optimal estimated-state-based policies in many of the test-bed POMDPs [1] popular in the RL literature. Nevertheless, we show a simple artificial POMDP in Figure 2 in which it has difficulty. This example will motivate our SarsaLandmark algorithm.

Consider learning memoryless (estimated-state is just the immediate observation) policies in the deterministic landmark-POMDP of Figure 2. In this figure, non-landmark states are represented by circles while landmark-states are denoted by squares. Rewards are shown on the edges corresponding to the actions. Only state *1* has multiple actions: *solid*, *dashed*, and *dotted*, to choose from. State *6* is a terminal state and all states have unique observations except for states *2* and *3* which have the same observation *obs2*. State *3* leads to a large negative reward (C) while state *2* leads to the only positive reward (r). States *2* and *3* produce the same observation, and because of this the Q-value of this observation will be determined by the relative frequency with which these two states are experienced. We will argue that for any $\lambda < 1$, it is possible to choose the value of r and C to make Sarsa(λ) converge to using the *dotted*-action in observation *obs1* instead of the optimal *solid*-action.

Note that the Q-value of the *dashed*-action will always be negative (or at least always be less than or equal to the value of the *solid*-action), while the Q-value of the *dotted*-action will always be zero. Therefore, in order for Sarsa(λ) to pick the *solid*-action, the Q-value(*obs 1*,*solid*-action) has to be positive. Sarsa(λ) with ϵ-greedy exploration for any fixed $\epsilon > 0$ will explore the *dashed*-action with probability at least $\epsilon/3$, and therefore we can make the value of observation *obs2* as negative as we want by making C as negative as needed. The Q-value of the *solid*-action in observation *1* will be a weighted combination of the negative value of observation *obs2* and the positive reward r. The smaller the λ, the more the weight on the negative value of observation *obs2*; conversely the larger the λ the more the weight on the positive r. Therefore, by making the value of *obs2* more and more negative we are forced to use larger and larger λ in order to make Sarsa(λ) work in this problem.

Table 4 experimentally confirms the above argument about the need for large λ. The first column specifies the value of r (smaller r means a more negative value for *obs2*), while the second column specifies the corresponding, empirically-found, minimum λ value below which the algorithm converges to the sub-optimal *dotted*-action. These experiments were performed with a fixed exploration rate of 30% for which we used a $C = -150$. The table empirically confirms that for any $\lambda < 1$ we can always choose a small enough $r > 0$ to make Sarsa(λ) find a sub-optimal policy. For smaller exploration rates, we would have to use a more negative value for C to get similar results.

Of course, Sarsa(1) will converge to the optimal *solid*-action for all settings of r and C in Figure 2. This is because the Monte-Carlo algorithm updates Q-values based solely on the discounted sum of rewards obtained and does not use the Q-value of *obs2* at all. Thus, Sarsa(1) will converge to the correct value, γr, for the *solid*-action. This then seems to imply that we should simply use Sarsa(1) to learn policies in POMDPs. But there is a catch; just as for MDPs Sarsa(1) scales poorly because of the high variance obtained by us-

ing the discounted sum of a long trajectory of rewards. In MDPs we address this issue of large variance by using $\lambda < 1$. We cannot adopt that solution for POMDPs because as our counterexample shows the bias introduced by the estimated Q-values need not go to zero over time (in particular this was the case for the Q-value of *obs2* in the counterexample).

What we would like is an algorithm that has a lower variance than Sarsa(1) but computes the same Q-values as Sarsa(1). An algorithm that "bridges" over or skips the non-landmark states and uses Q-values only of the landmark-states would accomplish this goal. SarsaLandmark, presented next, is such an algorithm.

5. SARSALANDMARK

The central idea behind SarsaLandmark is quite simple. We use non-decreasing eligibility traces, just as in Sarsa(1), except that whenever we encounter a landmark state, we drop the eligibility trace to zero (see bottom right-hand graph in Figure 1). Note that if it is applied to an MDP, SarsaLandmark is equivalent to Sarsa(0) and if applied to a POMDP with no landmark states (the terminal state is always a landmark state, however), it is equivalent to Sarsa(1). The update equations look like this: on experiencing transition $< x_t, a_t, r_t, x_{t+1} >$ at time step t, the following updates are performed in order:

$$\eta_t(x_t, a_t) = 1$$

If x_t is a landmark state, then

$$\eta_t(x, a) = 0, \ \ \forall\, (x \neq x_t \text{ or } a \neq a_t),$$

else

$$\eta_t(x, a) = \gamma \eta_{t-1}(x, a), \ \ \forall\, (x \neq x_t \text{ or } a \neq a_t)$$

and finally,

$$Q_{t+1}(x, a) = Q_t(x, a) + \alpha \eta_t(x, a) \delta_t, \ \ \forall\, x, a,$$

The net effect is that the Q-value of (x_t, a_t) gets updated based on the discounted sum of rewards until the first occurrence of a landmark state plus the appropriately discounted value of the landmark state. Thus, in the POMDP of Figure 2, SarsaLandmark will converge to the same value as Sarsa(1) because it too will bridge over *obs2* in computing the value of (*obs1*, *solid*-action).

For general landmark-POMDPs, it is clear from Figure 1 that for the same episodic experience, the eligibility traces of SarsaLandmark will never be longer than the eligibility traces of Sarsa(1) and will mostly be shorter. In other words, SarsaLandmark will update on the basis of a shorter trajectory than will Sarsa(1). This in turn means that SarsaLandmark will have a lower variance than Sarsa(1) (because the variances of the rewards along a trajectory add and hence the shorter the trajectory the smaller the variance). In fact, the more landmark states there are in a POMDP the lower will be the variance of SarsaLandmark relative to Sarsa(1). But of course, while the estimated Q-values of landmark states are inaccurate, the SarsaLandmark update will have a bias.

The situation is much like what we faced with decaying eligibility traces in the MDP setting. And just like in that setting the bias will go to zero as the Q-value estimates of the landmark states improve and SarsaLandmark will converge faster than Sarsa(1)[1].

6. ANALYSIS FOR POLICY EVALUATION

In this section, we analyze the performance of SarsaLandmark at *policy evaluation* in which the goal is to estimate the expected value of the discounted sum of payoffs obtained by executing a fixed policy. This is a subproblem of the more general problem of learning improved policies and is the same setting in which most of the analyses of eligibility-trace based algorithms have been done (e.g., [2, 5]). For ease of analysis, here we will only consider the case where the estimated-state is the immediate observation. Our results extend to the case where the estimated-state is a function of a bounded history window.

Let π be the stochastic memoryless policy we wish to evaluate. By a slight abuse of notation we will let $\pi(a|o)$ denote the probability of executing action a in observation o under policy π. Executing policy π in the POMDP will execute a policy π_M in the underlying MDP, where:

$$\pi_M(a|s) = \sum_o \pi(a|o) B(o|s).$$

Policy π_M will induce a Q-value function over state-action pairs (s, a) in the underlying MDP that we denote $Q^{\pi_M}(s, a)$ — this is the expected value of the discounted sum of payoffs received in the MDP by starting in state s, executing action a and following policy π_M thereafter. We also define the stationary conditional probability of the underlying state being s when the agent observes o while following policy π, as follows:

$$Prob(s|o, \pi) = \frac{O(o|s) Prob(s|\pi_M)}{Prob(o|\pi)},$$

where $Prob(s|\pi)$ is the stationary distribution over states under policy π_M, and $Prob(o|\pi)$ is the stationary distribution over observations under policy π.

Next, we derive a formal relationship between Q^{π_M} and the Q-value function, Q^π, that SarsaLandmark will converge to when following policy π.

THEOREM 1. *For any landmark-POMDP and for any memoryless policy π, under standard stochastic approximation conditions on the step-size ($\sum \alpha = \infty$ and $\sum \alpha^2 < \infty$), SarsaLandmark will converge asymptotically w.p.1. to a Q-value function Q^π, such that the following relationship holds: $\forall\, o \in O, a \in A$:*

$$Q^\pi(o, a) \ = \ \sum_{s \in S} Prob(s|o, \pi) Q^{\pi_M}(s, a) \qquad (2)$$

where $Prob(s|o, \pi)$ and Q^{π_M} are defined above.

Proof (sketch): The key observation is that SarsaLandmark updates never use estimated Q-values of any observation-action pair except when the observation is that of a landmark state. In particular, the updates of the Q-values of (landmark-state,action) proceed just as they would in the

[1]Recall that the main difficulty with using Sarsa($\lambda < 1$) is that the bias in the Q-value estimates of some estimated-states never goes to zero; SarsaLandmark avoids that fate by only using payoffs and the Q-values of landmark states to update the Q-values of estimated-states.

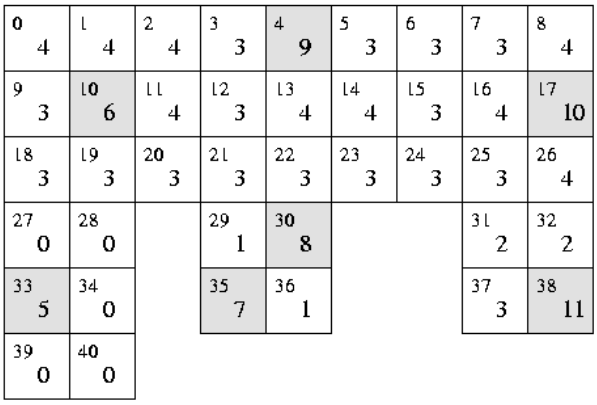

Figure 3: Small Landmark-POMDP. The state labels are on the upper left corner and the observation label is in the middle of the square. This example was inspired by McCallum's (1993) Cheese-Maze problem, except that we relabeled the observations to allow for a good memoryless policy. Inspection of the figure will confirm that the following memoryless policy [(0,Up),(1,U), (2,D),(3,R), (4,D),(5,U),(6,R),(7,U),(8,U),(9,R),(10,D)] is optimal.

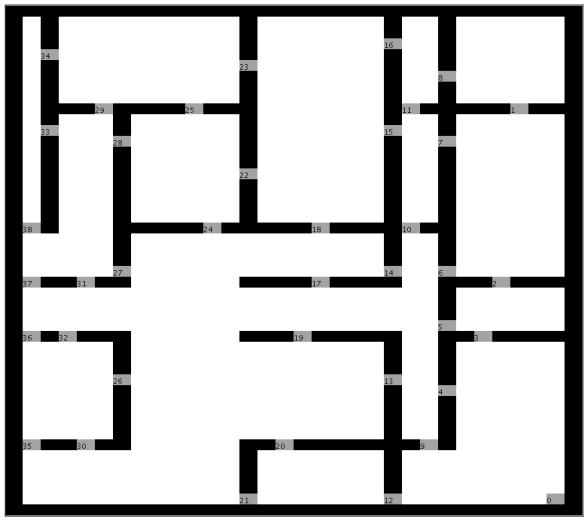

Figure 4: A larger rooms-world. The doorways between rooms are landmark states. The goal state is in the bottom right corner. The observations for the states inside the room are defined in the text.

underlying MDP (following the given policy), and thus converge to $Q^{\pi_M}(s_L, a)$ where s_L corresponds to the underlying MDP state for the given landmark state.

Given this convergence for landmark states and the fact that the Q-values for all other observation-action pairs are updated only on the Q-values of the landmark states, the updates of the other (o, a) pairs will mix in the updates of the underlying hidden state-action (s, a) pairs in proportion to $Prob(s|o, \pi)$.

COROLLARY 1. *The asymptotic Q-value of SarsaLandmark for any landmark state will be equal to the Q-value of the same state in the underlying MDP, i.e., if s is a landmark state with observation o, then*

$$Q^{\pi}(o, a) = Q^{\pi_M}(s, a)$$

COROLLARY 2. *For any landmark-POMDP and for any memoryless policy π, SarsaLandmark and Sarsa(1) or Monte-Carlo converge asymptotically to the same Q-value function of observation-action pairs.*

Corollary 2 is the main theoretical result of this paper and follows immediately from Theorem 1 and from the fact that Monte-Carlo is already known to converge to the RHS of Equation 2 [3].

7. EMPIRICAL RESULTS FOR LEARNING CONTROL

Here we empirically test the performance of SarsaLandmark on a few landmark-POMDPs.

7.1 Small deterministic landmark-POMDP

Figure 3 shows our first test problem. It was designed to have a memoryless policy that is optimal, and our goal is to check whether SarsaLandmark will find the optimal policy and whether the Q-values of landmark states will converge to their true Q-values in the underlying MDP (as predicted by the theoretical analysis).

Figure 3 shows a grid world, with four actions (move up, down, left, and right) available in each state. The actions are deterministic and the result of an illegal action is just to remain in the same state. For each state, the numbers in the upper left corner show the state in the underlying MDP. The numbers in the middle show the observation received by the agent. There are seven landmark states that are highlighted in the figure by shaded squares, and that have observation numbers *5* to *11*. Landmark state *11* is a goal/terminal state. Rewards are -1 for all transitions, except for the transition into the terminal state, for which the reward is 20. The discount factor is 0.9.

SarsaLandmark is an on-policy algorithm, and thus the Q-values learned depend on the actual exploratory policy being followed. To verify convergence to an optimal policy, we could use a constant but small exploration rate, but in order to verify that the Q-values learned for landmark states converge to the Q-values of the same states in the underlying MDP, we need to reduce the exploration rate over time. In particular, we used ϵ-greedy SarsaLandmark with a decreasing exploration rate that started at 0.05, and was decreased by 0.00001 with each action taken. We were able to confirm for a set of different step-size schedules, that the learned policy converged to an optimal memoryless policy and the Q-values of the landmark states converged to those of the same states in the underlying MDP.

7.2 Larger landmark-POMDPs

In the preceding problem, the existence of a good memoryless policy allowed us to use the immediate observations as estimated state. How will SarsaLandmark fare in problems that require the use of memory in estimating state? To test

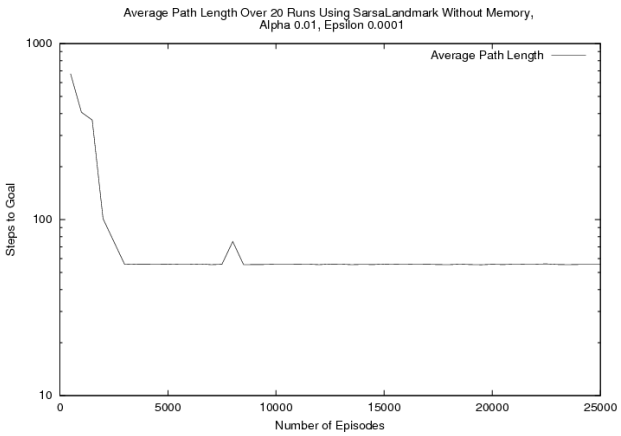

Figure 5: Average number of steps to the goal in the rooms-world of Figure 4 as a function of number of training episodes. Note the log-scale on the y-axis.

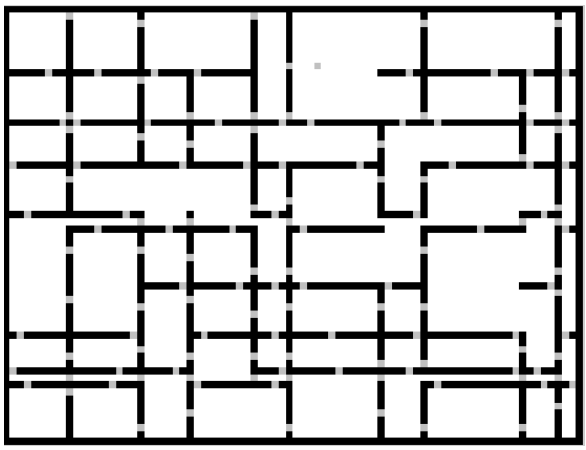

Figure 6: The largest rooms-world. The doorways are landmark states. The goal state is the gray square in the middle of the large room at the top. The observations for the states inside the rooms are described in the text along with other details.

this as well as to test how SarsaLandmark scales to larger POMDPs, we constructed the two rooms-world problems shown in Figures 4 and 6.

Each rooms-world consist of many differently sized and shaped rooms. Doorways between rooms are landmarks and thus produce unique observations. When not in a landmark state, the agent receives an observation based on the distance to the walls in each of the four cardinal directions. Each distance is categorized into one of four classes: next to the wall, one step from the wall, closer to the wall than to the wall in the opposite direction, and further from the wall than to the wall in the opposite direction. Therefore, there are roughly $4^4 = 256$ non-doorway observations. There are four actions that attempt to move the agent north, south, east and west. The actions succeed only with probability 80% and failure results in no movement. An action that would move the agent into a wall also results in no movement. In both domains, there is a single terminal/goal state and a transition into it results in a reward of 20; all other transitions have zero reward. The discount factor is 0.99.

The following training and testing method was used with both domains. Every training episode was started on a randomly chosen landmark state. An episode ended when the agent reached the terminal state. The agent's performance was tested after every 500 training episodes; a test consisted of starting the agent on each landmark state, and then measuring the average number of steps to the goal when following a greedy policy. To keep things bounded a test was cut off if the agent took 5000 steps without reaching the goal.

Next we present results for the smaller of the two rooms-worlds (Figure 4). It is defined on a grid of size 30 by 45 with more than 1200 underlying states. There are 39 landmark observations. The goal state is located in the lower right corner. As a point of comparison to our results below, we note that the optimal policy on the underlying MDP takes an average of 47.86 steps to reach the goal.

First, we tested the performance of constant step-size and exploration rate SarsaLandmark using just the immediate observations as estimated state. A range of values of α and ϵ were tried and the results were found to be robust to reasonable choices for these two parameters. Figure 5 presents results for $\alpha = 0.01$ and $\epsilon = 0.0001$. It shows the number of steps to the goal averaged over starting in each doorway and over 20 runs as a function of the number of training episodes.

The average number of steps to the goal settles at about 56.25 steps after about 3500 episodes. The 56.25 step performance is surprisingly close to the 47.86 step performance of the optimal MDP policy. Further analysis showed that this was a result of the goal state being in the bottom right corner and so a policy that generally heads in that direction seems to work reasonably well. In fact, in this problem using memory in the estimated state was found not to help performance.

In order to ensure that the use of memory in estimating state will help in the next rooms-world (Figure 6), we choose the goal state to be in the middle of the grid (it is the gray square in the middle of the large upper room). The size of the grid is 60 by 80 and while the total number of states (about 4000) is therefore not large relative to what has been handled in the case of MDPs, learning in a POMDP of even this size is infeasible with traditional belief-state approaches. As a point of reference, the average number of steps to the goal in the underlying MDP is 60.6 steps.

Again, we first tried SarsaLandmark with just the observations as estimated state. The training methodology was exactly the same as for the previous rooms-world. In this case, however, SarsaLandmark was simply unable to find a policy that gets to the goal reliably from all doorways. This is not surprising given the layout of the domain.

As a simple extension of the SarsaLandmark algorithm, we added memory to the estimated state; specifically, we allowed the agent to remember the last landmark state it had seen. The results for $\alpha = 0.002$ and $\epsilon = 0.001$ are shown in Figure 7. It shows the number of steps to the goal averaged over starting in all doorways and over 20 runs as a function of the number of training episodes. The figure

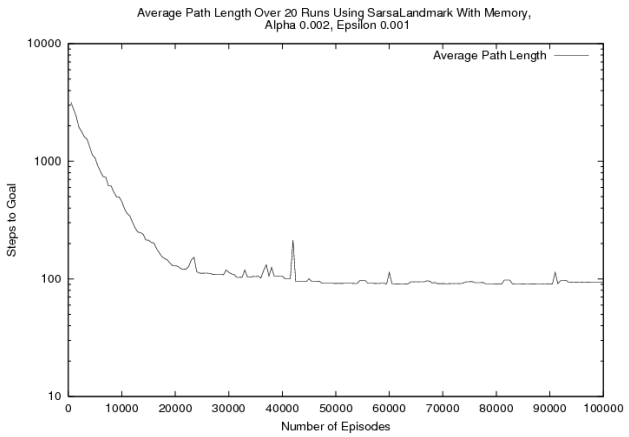

Figure 7: Average number of steps to goal for the rooms-world of Figure 6 as a function of number of training episodes. Note the log scale on the y-axis.

shows that in about 35,000 episodes the average number of steps to the goal settles to about 93.75. This is quite good given the fact that the estimated state still leaves the agent with only partial observability. The problem is too large for us to be able to otherwise confirm that this is indeed the best the agent can do in the class of policies representable as a mapping from its particular estimated-state to actions.

8. DISCUSSION

In this paper, we presented SarsaLandmark, an algorithm that uses a variable length, non-decaying, eligibility-trace to solve landmark POMDPs. We showed that it can solve POMDPs that Sarsa($\lambda < 1$) cannot. We also provided evidence that SarsaLandmark can be viewed as a lower-variance version of Monte-Carlo or Sarsa(1).

9. REFERENCES

[1] A. Cassandra. Tony's pomdp page, 1999. `http://www.cs.brown.edu/research/ai/pomdp/index.html`.

[2] T. Jaakkola, M. I. Jordan, and S. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201, 1994.

[3] T. Jaakkola, S. P. Singh, and M. I. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. In G. Tesauro, D. S. touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 345–352. Morgan Kaufmann, 1995.

[4] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[5] M. Kearns and S. Singh. Bias-variance error bounds for temporal difference updates. In *COLT Proceedings*. Morgan Kaufmann, 2000.

[6] J. Loch and S. Singh. Using eligibility traces to find the best memoryless policy in partially observable Markov decision processes. In *Proc. 15th International Conf. on Machine Learning*, pages 323–331. Morgan Kaufmann, San Francisco, CA, 1998.

[7] R. A. McCallum. Overcoming incomplete perception with utile distinction memory. In P. Utgoff, editor, *Machine Learning: Proceedings of the Tenth International Conference*, pages 190–196. Morgan Kaufmann, 1993.

[8] T. Perkins and M. Pendrith. On the existence of fixed points for Q-learning and Sarsa in partially observable domains. In *ICML*, 2002.

[9] T. Perkins and D. Precup. A convergent form of approximate policy iteration. In *NIPS 2002*, 2003.

[10] S. Singh, T. Jaakkola, and M. I. Jordan. Learning without state-estimation in partially observable Markovian decision processes. In W. W. Cohen and H. Hirsh, editors, *Machine Learning: Proceedings of the Eleventh International Conference*, pages 284–292. Morgan Kaufmann, 1994.

[11] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[12] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[13] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge Univ., Cambridge, England, 1989.