Using Eligibility Traces to Find the Best Memoryless Policy in Partially Observable Markov Decision Processes

John Loch
Department of Computer Science
University of Colorado
Boulder, CO 80309-0430
loch@cs.colorado.edu

Satinder Singh
Department of Computer Science
University of Colorado
Boulder, CO 80309-0430
baveja@cs.colorado.edu

Abstract

Recent research on hidden-state reinforcement learning (RL) problems has concentrated on overcoming partial observability by using memory to estimate state. However, such methods are computationally extremely expensive and thus have very limited applicability. This emphasis on state estimation has come about because it has been widely observed that the presence of hidden state or partial observability renders popular RL methods such as Q-learning and Sarsa useless. However, this observation is misleading in two ways: first, the theoretical results supporting it only apply to RL algorithms that do not use eligibility traces, and second these results are worst-case results, which leaves open the possibility that there may be large classes of hidden-state problems in which RL algorithms work well without any state estimation.

In this paper we show empirically that Sarsa(λ), a well known family of RL algorithms that use eligibility traces, can work very well on hidden state problems that have good memoryless policies, i.e., on RL problems in which there may well be very poor observability but there also exists a mapping from immediate observations to actions that yields near-optimal return. We apply conventional Sarsa(λ) to four test problems taken from the recent work of Littman, Littman Cassandra and Kaelbling, Parr and Russell, and Chrisman, and in each case we show that it is able to find the best, or a very good, memoryless policy without any of the computational expense of state estimation.

1 Introduction

Sequential decision problems in which an agent's sensory observations provide it with the complete state of its environment can be formulated as Markov decision processes, or MDPs, for which a number of very succesful planning (Sutton & Barto, 1998) and reinforcement learning (Barto et al., 1983; Sutton, 1988; Watkins, 1989) methods have been developed. However, in many domains, e.g., in mobile robotics, and in multi-agent or distributed control environments, the agent's sensors at best give it partial information about the state of the environment. Such agent-environment interactions suffer from hidden-state (Lin & Mitchell, 1992) or perceptual aliasing (Whitehead & Ballard, 1990; Chrisman, 1992) and can be formulated as *partially observable Markov decision processes*, or POMDPs (e.g., Sondik, 1978). Therefore, finding efficient reinforcement learning methods for solving interesting sub-classes of POMDPs is of great practical interest to AI and engineering.

Recent research on POMDPs has concentrated on overcoming partial observability by using memory to estimate state (Chrisman, 1992; McCallum, 1993; Lin & Mitchell, 1992) and on developing special purpose planning and learning methods that work with the agent's state of knowledge, or belief state (Littman et al., 1995). In part, this emphasis on state estimation has come about because it has been widely observed and noted that the presence of hidden state renders popular and succesful reinforcement learning (RL) methods for MDPs, such as Q-learning (Watkins, 1989) and Sarsa (Rummery & Niranjan, 1994), useless on POMDPs (e.g., Whitehead, 1992; Littman, 1994; Singh et al., 1994). However, this observation is misleading in two ways: first, the theoretical results (Singh et al., 1994; Littman, 1994) supporting it only apply to RL algorithms that do not use eligibility traces, and second, these results are worst-case results which leaves open the possibility that there may be large classes of POMDPs in which existing RL algorithms work well without any state estimation.

The main contribution of this paper is to show empirically that Sarsa(λ), a well known family of reinforcement learning algorithms that use eligibility traces, can work very well on POMDPs that have good memoryless policies, i.e., on problems in which there may well be very poor observability but there also exists a mapping from the agent's immediate observations to actions that yields near-optimal return. We also show how this can be extended to low-order-memory-based policies. This contribution is significant, because it may be that most real-world engineering problems that are well designed have good memoryless or good low-order-memory-based policies. We apply conventional Sarsa(λ) on four test problems taken from recent published work on POMDPs and in each case show that it is able to find the best, or a very good, memoryless policy without any of the computational expense of state estimation. However, these results have to be interpreted with caution for the problem of finding optimal memoryless policies in POMDPs is known to be computationally challenging (Littman, 1994); they are evidence that Sarsa(λ) is at least competitive to and at best better than other existing algorithms for solving POMDPs when good low-order-memory-based policies exist.

2 POMDP Framework

In this section we briefly describe the POMDP framework. An environment is defined by a finite set of states S, the agent has recourse to a finite set of actions A, and the agent's sensors provide it observations from a finite set X. On executing action $a \in A$ in state $s \in S$ the agent receives expected reward r_s^a and the environment transits to a random state $s' \in S$ with probability $P_{ss'}^a$. The probability of the agent observing $x \in X$ given that the environment's state is s is $O(x|s)$. In the reinforcement learning (RL) problem the agent does not know the transition and observation probabilities P and O and its goal is to learn an action selection strategy that maximizes the *return*, i.e. the expected discounted sum of rewards received over an infinite horizon, $E\{\sum_{t=0}^{\infty} \gamma^t r_t\}$, where $0 \leq \gamma < 1$ is the discount factor that makes immediate reward more valuable than reward more distant in time, and r_t is the reward at time step t.

In fully observable RL problems or MDPs it is known that there exists an optimal policy that is memoryless, i.e., is a mapping from states to actions, $S \rightarrow A$. RL algorithms such as Q-learning and Sarsa are able to provably find such memoryless optimal policies in MDPs. It is known that in POMDPs the best memoryless policy can be arbitrarily suboptimal in the worst case (Singh et al., 1994). We ask below if these same RL algorithms can find the best memoryless policy in POMDPs (Jaakkola et al., 1995; Littman, 1994), regardless of how good or how bad it is; for if they are able to find it, then they can at least be useful in POMDPs with good memoryless policies. We note that the success of RL algorithms when using compact function approximation in fully observable problems (Barto et al., 1983; Tesauro, 1995) provides some evidence that this is possible because the use of compact function approximation introduces hidden state into otherwise completely observable MDPs.

3 Eligibility Traces and Sarsa(λ)

In MDPs reinforcement learning algorithms such as Sarsa(λ) use experience to learn estimates of optimal Q-value functions that map state-action pairs, s, a, to the optimal return on taking action a in state s. The transition at time step t, $< s_t, a_t, r_t, s_{t+1} >$, is used to update the Q-value estimate of all state-action pairs in proportion to their eligibility. The idea behind the eligibilities is very simple. Each time a state-action pair is visited it initiates a short-term memory or trace that then decays over time (exponentially with parameter $0 \leq \lambda \leq 1$). The magnitude of the trace determines how eligible a state-action pair is for learning. So state-action pairs visited more recently are more eligible.

In POMDPs the transition information available to the agent at time step t is $< x_t, a_t, r_t, x_{t+1} >$. A straightforward way to extend RL algorithms to POMDPs is to learn Q-value functions of observation-action pairs, i.e., to simply treat the agent's observations as states. Below we describe standard Sarsa(λ) applied to POMDPs in this manner. At step t the Q-value function is denoted Q_t and the eligibility trace function is denoted η_t. On experiencing transition $< x_t, a_t, r_t, x_{t+1} >$ the following updates are performed in order:

$$\begin{aligned} \eta_t(x_t, a_t) &= 1 \\ \forall\ (x \neq x_t \text{ or } a \neq a_t);\ \eta_t(x, a) &= \gamma\lambda\eta_{t-1}(x, a) \end{aligned}$$

$\forall x$ and a;

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

where $\delta_t = r_t + \gamma Q_t(x_{t+1}, a_{t+1}) - Q_t(x_t, a_t)$, and α is the step-size. The eligibility traces are initialized to zero, and in episodic tasks they are reinitialized to zero after every episode. The greedy policy at time step t assigns to each observation x the action $a = argmax_b Q_t(x, b)$. Note that the greedy policy is memoryless.

3.1 Using Sarsa(λ) with Observation Histories

The Sarsa(λ) algorithm can also be easily used to develop memory-based policies by simply learning a Q-value function over estimated-states and actions, and by keeping eligibility traces for estimated-state and action pairs. So for example, we could augment the immediate observation with the past K observations to form the estimated-state and derive a memory-based policy that maps $K+1$ observations to actions. The only change to the equations in (1) would be that the immediate observations (x's) would be replaced by the estimated states.

4 Empirical Results

The Sarsa(λ) algorithm was applied in an identical manner to four POMDP problems taken from the recent literature and described below. Here we describe the aspects of the empirical results common to all four problems. At each step, the agent picked a random action with a probability equal to the exploration rate, and a greedy action otherwise. Except where explicitly noted, we used an initial exploration rate of 20% decreasing linearly with each action (step) until the 200000^{th} action from where onwards the exploration rate was 0%. Q-values were initialized to 0. The agent starts each episode in a problem specific start state or a randomly selected start state as specified by the originators of the problems. Both the step-size (α) and the λ values are held constant in each experiment. We did a coarse search over α and λ for each problem but present results only for $\lambda = 0.9$ and $\alpha = 0.01$ which gave about the best performance across all problems. In all cases, a value of λ between 0.8 and 0.975 worked the best. This is qualitatively similar to the results obtained for MDPs, and a bit surprising given that Sarsa(1) (or Monte-Carlo) has been recommended as the way to deal with hidden state (Singh et al., 1994).

The data for the learning curves is generated as follows: after every 1000 steps (actions) the greedy policy is evaluated offline to generate a problem specific performance metric. All the learning-curves below are plotted after smoothing this data by doing a running average over 30 data points.

For each POMDP we first present its structure by defining the states, actions, rewards, and observations and then we present our results.

4.1 Sutton's Grid World

Sutton's grid world problem (see Figure 1A) is from Littman (1994) who took a navigation gridworld from Sutton (1990) and made it a POMDP by not allowing its exact position to be known to the agent.

States: This POMDP is a 9 by 6 grid with several obstacles and a goal in the upper right corner (see Figure 1A). The state of the environment is determined by the grid square the agent occupies. State transitions are deterministic.

Actions: The agent can choose one of 4 actions: move north, move south, move east, and move west.

Observations: The agent can observe its 8 neighboring grid squares yielding 256 possible observations. Only 30 (of the 256 possible) unique observations occur in the gridworld. Observations are deterministic. Figure 1A shows the gridworld with observations indicated by the number in the lower right corner of each square.

Rewards: The agent receives a reward of -1 for each action that does not transition to the goal state. A reward of 0 is received for any action leading to the goal state.

When the agent reaches the goal state it transitions to a uniformly random start state.

4.1.1 Sarsa(λ) Results

After every 1000 steps of experience in the world, the greedy policy is evaluated to determine the total number of steps required to reach the goal from every possible non-goal start state (46 start states). The agent is limited to a maximum of 1000 steps to reach the goal. Thus a policy which cannot reach the goal from any start state would have a total steps to goal of $46,000$.

Sarsa(λ) converged to the 416 total step policy shown with arrows in Figure 1A; the learning-curve is shown in Figure 1B. The total steps to the goal for the optimal policy in the underlying MDP is 404, and so in this case a very good memoryless policy was found. This 416 step policy matches exactly with the 416 step policy Littman (1994) found using an expensive branch

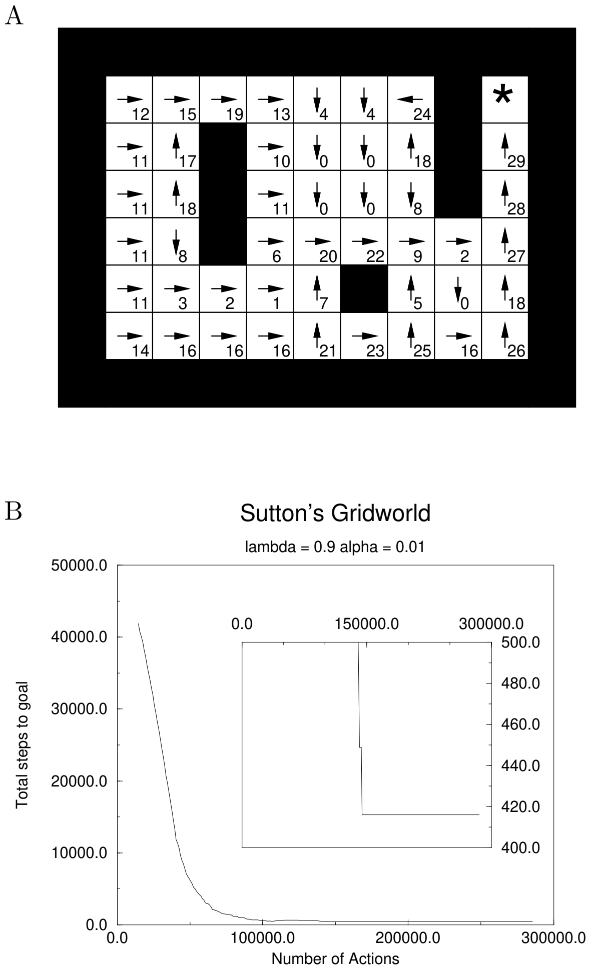

Figure 1: Sutton's Grid World (from Littman, 1994). A) The grid world environment. The numbers on the lower right are the observations. The arrows show the optimal memoryless policy found by Sarsa. B) The total steps to goal of the greedy policy as a function of the amount of learning steps. The inset plot shows the same data at a different scale.

and bound method that searches directly in memoryless policy space and is guaranteed to find the optimal memoryless policy. Note that the number of possible memoryless policies is (4 actions, 30 observations) $4^{30} = 1.2 \times 10^{18}$ policies.

Observe that in Figure 1A the agent learns to go left in the state just to the left of the goal. This is because it has to go up in the state immediately below (observation 18) because of its aliasing with the state 4 steps below the goal (both states have 3 walls to the right).

4.2 Littman, Cassandra, and Kaelbling's 89 State Office World

States: The gridworld for Littman et al.'s (1995) 89 state office problem is shown in Figure 2A. The state of the environment is the combination of the grid square that the agent is occupying and the direction that the agent is facing (N, S, E, W). The are 22 possible agent locations times 4 directions for 88 states plus the goal state for 89 total states. State transitions are stochastic.

Actions: The agent can choose one of 5 actions: stay in place, move forward, turn right, turn left, and turn around. Both the state transitions and the observations are noisy with the agent getting the correct observation only 70% of the time.

Observations: The agent can observe the relative position of obstacles in 4 directions: front, back, left and right. There are 16 possible observations plus the goal observation.

Rewards: The agent receives a reward of +1 for any action leading to the goal observation with all other rewards equal to 0.

After reaching the goal observation the agent transitions to a uniformly random start state.

4.2.1 Sarsa(λ) Results

After every 1000 steps of experience the greedy policy is evaluated. As in Littman et al., for each evaluation 251 trials are run using the greedy policy with a maximum step cutoff at 251 steps. Two performance metrics are used: the median number of steps to the goal for the 251 trials, and the percent of the 251 trials which reach the goal state within 251 steps.

The best memoryless policy found by Sarsa(0.9) was able to reach the goal on average 77% of the 251 trials (see Figure 2B) with a median number of steps to goal of 73 steps (see Figure 2C). The best policy

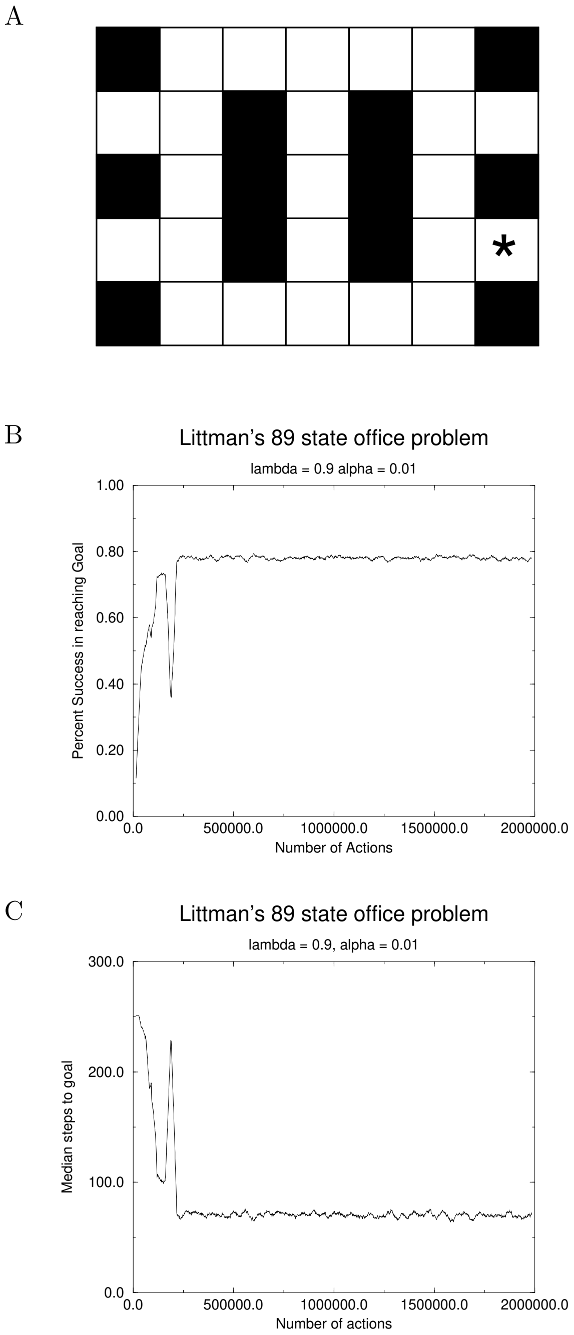

Figure 2: Littman et al.'s 89 state office world. A) The office world environment where the goal state is denoted with a star. The state of the environment is the combination of the grid square that the agent is occupying and the direction that the agent is facing (N, S, E, W). B) The percentage of trials with the greedy policy that succeed in getting to the goal in less than 251 steps. C) Median number of steps to goal of the greedy policy as a function of the number of learning steps.

found by Sarsa(0.9) outperformed all of the memory-based policies found by Littman et al. in their Table 3. Their best policy was able to reach the goal in only 44.6% of the 251 trials with a median steps to goal of $>$ 251 steps and was found using truncated value iteration algorithm on belief states. Littman et al. also presented a hybrid method that finds a policy that reached the goal in 58.6% of the trials (still below the percentage for the best memoryless policy found by Sarsa(0.9) with median steps to goal of 51 steps (this is better than Sarsa(0.9)'s 73 steps).

There are $5^{16} = 1.53 \times 10^{11}$ possible memoryless policies for this problem. Therefore it is not practical to enumerate the performance of every possible policy to verify if the policy found by Sarsa(0.9) is indeed the optimal memoryless policy, but its performance vis-a-vis the state-estimation based methods of Littman et al. was encouraging.

4.3 Parr and Russell's Grid World

States: Parr and Russell's (1995) gridworld consists of 11 states in a 4 by 3 grid with a single obstacle (see Figure 3A). The state of the environment is determined by the grid square occupied by the agent.

Actions: The agent can choose one of 4 actions: move north, move south, move east, and move west. State transitions are stochastic with the agent moving in the desired direction 80% of the time and slipping to either side 10% of the time.

Observations: The agent can only observe if there is a wall to its immediate left or right. There are 4 possible observations corresponding to the combinations of left and right obstacles plus two observations for the goal and penalty states yielding a total of 6 observations. Observations are deterministic.

Rewards: There is a goal state in the upper right corner with a penalty state directly below the goal state. The agent receives a reward of -0.04 for every action which does not lead to the goal or penalty state. The agent receives a reward of $+1$ for any action leading to the goal state and a reward of -1 for any action leading to the penalty state.

4.3.1 Sarsa(λ) Results

Every 1000 steps the greedy policy was evaluated and the learning curve is presented in Figure 3B. The average reward per step was computed for 101 trials of up to 101 steps per trial. There are $4^6 = 4096$ possible memoryless policies for this problem. We veri-

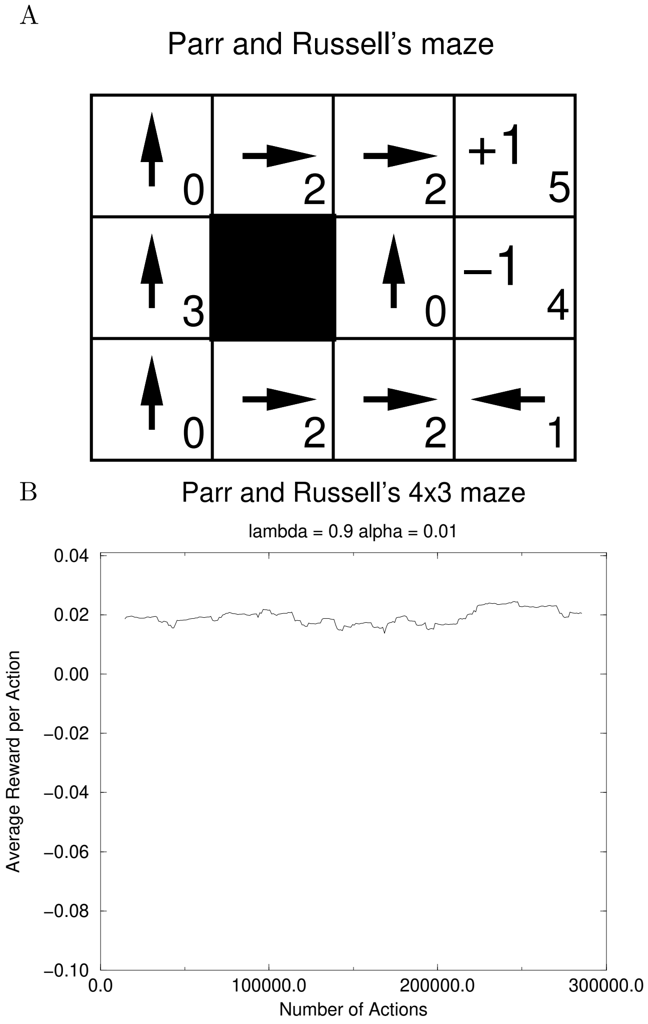

Figure 3: Parr & Russell's Grid World. A) The grid-world environment. The numbers in the lower right are the observations. The arrows show the optimal memoryless policy found by Sarsa. B) The average reward per action of the memoryless greedy policy as a function of the number of learning steps.

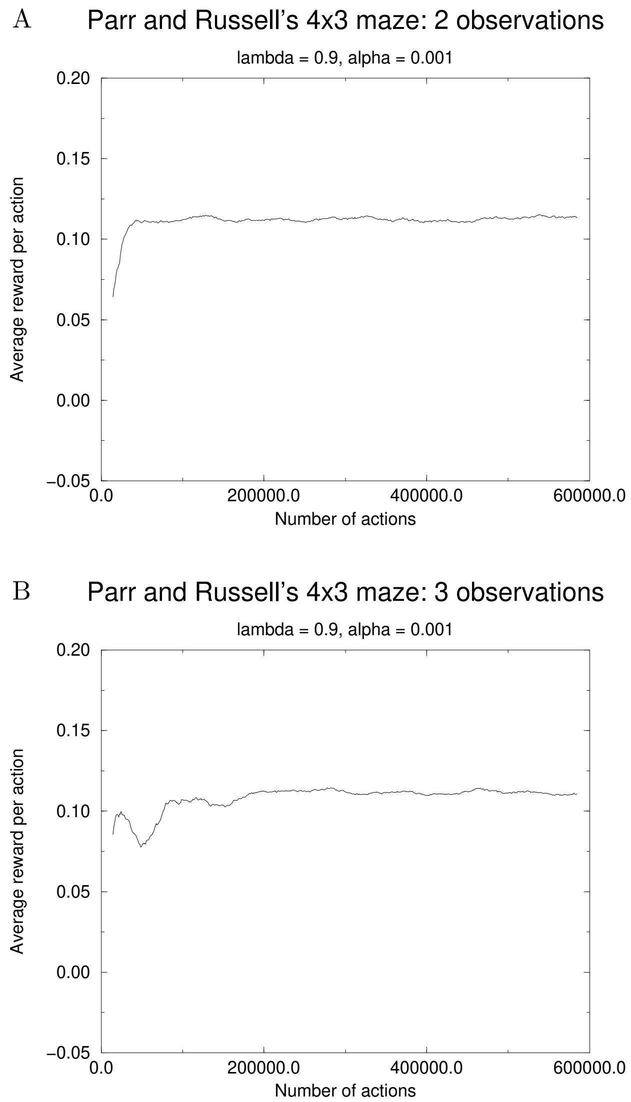

Figure 4: Parr & Russell's Grid World. A) We add one past observation to the immediate observation. The performance of the greedy policy. B) We add two past observations to the immediate observation. The performance of the greedy policy. Note the different scales.

fied that Sarsa(0.9) found the optimal memoryless policy by evaluating the performance of all 4096 possible policies. In this problem, the best memoryless policy is rather poor compared to policies which use memory. The best memoryless policy yields an average reward per step of 0.024 compared to the memory-based policy found by the Witness algorithm (Littman et al., 1995) which yields an average reward per step of 0.1108.

Parr & Russell's SPOVA-RL (Smooth Partially Observable Value Approximation Reinforcement Learning) algorithm learns a value function over belief states and did even better yielding an average reward per step of 0.12 with a memory-based policy[1].

The poor relative performance of the optimal memoryless policy is due to the non-optimal actions the agent must take in the aliased states. For example observation 0 (see Figure 3A) is observed for 3 states in the grid. The state to the left of the penalty state is observed as observation 0 and causes the optimal action in observation 0 to be move north instead of move east. This causes the agent to continuously bump into the upper left corner wall until the transition noise causes a transition to the state to the east.

We investigated the performance improvement obtained by Sarsa(λ) when the immediate observation is augmented with 1 and with 2 previous observations. The performance of the policy using 1 previous observation yielded an average reward per step of 0.1124 (see Figure 4A) which is better than the policy found by the Witness algorithm and almost as good as the policy found by SPOVA-RL. Sarsa(λ) required fewer than 60 CPU seconds to find its policy compared to the 42 CPU minutes for SPOVA-RL and the 12 CPU hours required by the Witness algorithm (Parr & Russell, 1995). The 3-observation performance is shown in Figure 4B and is the same as the 2-observation performance.

We were able to verify that the policy found by Sarsa(λ) using 1 previous observation was indeed the optimal policy in that space. Only ten 2-observation sequences are encountered in the gridworld leading to $4^{10} = 1,048,576$ possible 2 observation policies. We evaluated the performance of all possible 2-observation policies and again verified that the policy found by Sarsa(λ) was the same as the best 2-observation policy.

[1] Parr and Russell state that their implementation of the Witness algorithm did not converge on this problem, which probably accounts for the better performance of SPOVA-RL relative to the exact Witness algorithm.

4.4 Chrisman's Shuttle Problem

States: Chrisman's (1992) shuttle problem involves an agent operating in an environment with 8 states, 3 actions, and 5 observations. The scenario consists of two space stations with loading docks. The task is to transport supplies between the two docks. There is noise in both the state transitions and observations.

Actions: The agent can execute one of 3 actions: go forward, backup, and turn around.

Observations: The 5 observations are: can see the least recently visited (LRV) station; can see the most recently visited (MRV) station; can see that we are docked in most recently visited (MRV) station; can see that we are docked in least recently visited (LRV) station; and can see nothing. There is sensor noise causing the agent to make faulty observations.

Rewards: The agent receives a reward of +10 when it docks with the least recently visited station. The agent must back into the dock to dock with the station. If the agent collides with the station by moving forward it receives a reward of -3. All other action rewards are 0.

4.4.1 Sarsa(λ) Results

Every 1000 steps (actions) the performance of the greedy policy is evaluated. The performance metric is the average reward per step for 101 trials of up to 101 steps (actions) each. There are (3 actions, 5 observations) $3^5 = 243$ possible memoryless policies. Sarsa(0.9) finds a memoryless policy which yields an average reward per step of 1.02 (see Figure 5A for the learning curve). We verified that the policy found by Sarsa(0.9) was indeed the optimal memoryless policy by evaluating the performance of the 243 possible memoryless policies.

The two best memory-based policies for Chrisman's shuttle problem found by Littman et al. (1995) were found through truncated exact value iteration and their Qmdp method. Truncated exact value iteration found a policy with an average reward per step of 1.805 while Qmdp yielded 1.809. The performance of the optimal memoryless policy is rather poor compared to the performance of policies using memory. This is due to the conservative nature of the optimal memoryless policy which avoids any forward actions so as to avoid receiving the -3 penalty for hitting the station while moving forward.

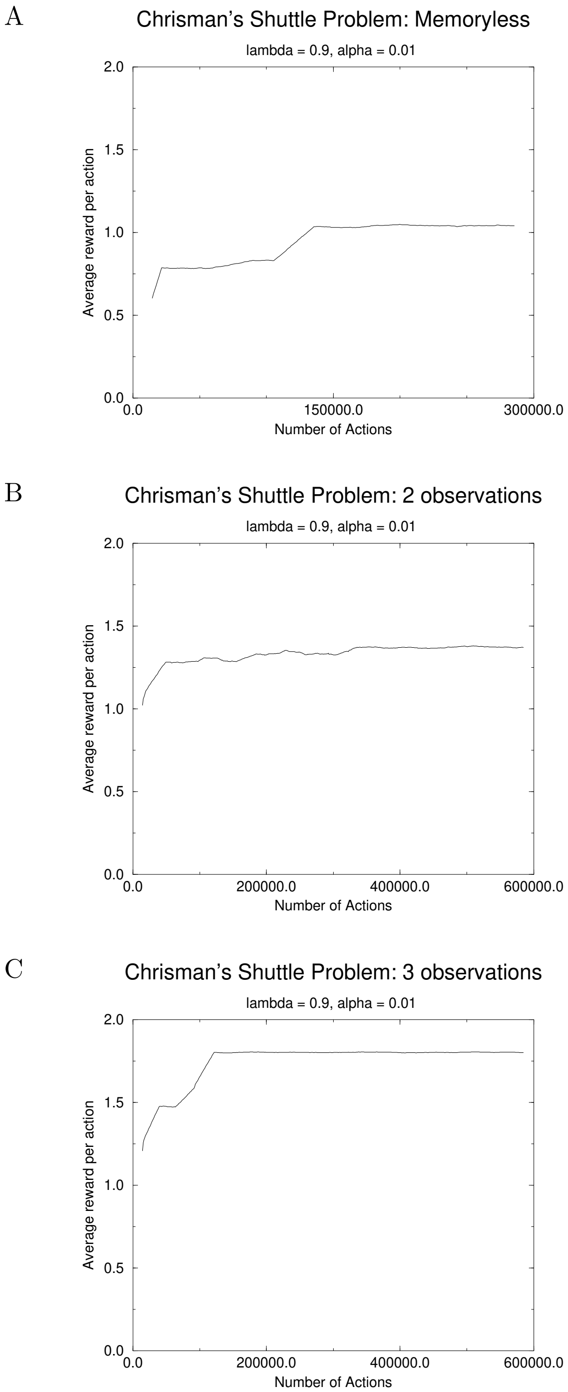

Figure 5: Chrisman's shuttle problem. A) The average reward per action of the memoryless greedy policy as a function of the number of learning steps. B) We add one past observation to the immediate observation. The performance of the greedy policy. C) We add two past observations to the immediate observation. The performance of the greedy policy.

We also investigated the performance improvement obtained by augmenting the current observation with 1 and 2 previous observations. By including the previous observation the performance improved by 37% to an average reward per step of 1.37 (see Figure 5B). By including the 2 previous observations the performance improved by 80% to an average reward per step of 1.804 (see Figure 5C). The performance of the best policy found by Sarsa with 2 previous observations is as good as the truncated exact value iteration method and the Qmdp method, again at a much lower computational cost.

4.5 Discussion

In all the empirical results presented above either we were able to confirm by enumeration that Sarsa found the best policy representable as a mapping from estimated-states (immediate, or immediate and past 1 or past 2 observations) to actions, or in cases where it was not possible to enumerate we observed that Sarsa did as well as the algorithms presented by the originators of the specific POMDPs. Speculating from these empirical results, we conjecture that Sarsa(λ) may be hard to beat in problems where there exists a good policy that maps the observation space to actions.

4.5.1 Why do Eligibility Traces Work?

Consider the set of states that map onto the same observation x. The neighbours of this set of states for some action a may map to several different observations. This can lead to conflicting pulls for the Q-value of x, a depending on which state is providing the experience; some may suggest a is good, some may suggest that a is bad. However these different pulls could get resolved if we considered what happens after n steps. Indeed if we wait until we get to the goal (Monte-Carlo or Sarsa(1)) there would be no confusion due to the hidden state at all. Eligibility traces allow an observation-action pair to access what happens many time steps later, bridging the gap to unambiguous information about the quality of an action. This reasoning indicates that there may be a minimum problem-specific λ that would be needed to bridge the smallest such "gap" in each problem. Our observations during the current work support this; however a careful analysis remains as future work.

5 Conclusion

Partial observability is inevitable in many sequential decision problems of interest to both AI and engineering. Given the worst-case computational intractability of POMDPs, it is useful to identify sub-classes of POMDPs and algorithms that work well in them. We believe that eligibility trace based RL methods such as Sarsa(λ) can be be useful in POMDPs that have good memoryless or good low-order-memory-based policies. We demonstrated this empirically on four POMDP problems from the recent literature. A more powerful result that remains future work would be to prove this theoretically.

Acknowledgements

Satinder Singh was supported by NSF grant IIS-9711753. We thank Michael Littman and the anonymous reviewers for many valuable comments.

References

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, *13*, 835–846.

Chrisman, L. (1992). Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *AAAI-92*.

Jaakkola, T., Singh, S., & Jordan, M. I. (1995). Reinforcement learning algorithm for partially observable Markov decision problems. In *Advances in Neural Information Processing Systems 7*, pages 345–352. Morgan Kaufmann.

Lin, L. J. & Mitchell, T. M. (1992). Reinforcement learning with hidden states. In *In Proceedings of the Second International Conference on Simulation of Adaptive Behavior: From Animals to Animats*.

Littman, M., Cassandra, A., & Kaelbling., L. (1995). Learning policies for partially observable environments: Scaling up. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 362–370, San Francisco, CA. Morgan Kaufmann.

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*.

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

Parr, R. & Russell, S. (1995). Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Rummery, G. A. & Niranjan, M. (1994). On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Dept.

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

Sondik, E. J. (1978). The optimal control of partially observable Markov processes over the infinite horizon: discounted case. *Operations Research*, *26*, 282–304.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*, 9–44.

Sutton, R. S. (1990). Integrating architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proc. of the Seventh International Conference on Machine Learning*, pages 216–224, San Mateo, CA. Morgan Kaufmann.

Sutton, R. S. & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Tesauro, G. J. (1995). Temporal difference learning and td-gammon. *Communications of the ACM*, *38*(3), 58–68.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. PhD thesis, Cambridge Univ., Cambridge, England.

Whitehead, S. D. & Ballard, D. H. (1990). Active perception and reinforcement learning. In *Proc. of the Seventh International Conference on Machine Learning*, Austin, TX. M.