Approximating Optimal Policies for Partially Observable Stochastic Domains

Ronald Parr, Stuart Russell
Computer Science Division
University of California
Berkeley, CA 94720, USA
{parr,russell}@cs.berkeley.edu
Tel: (510) 642-2038, Fax: (510) 642-5775

Abstract

The problem of making optimal decisions in uncertain conditions is central to Artificial Intelligence. If the state of the world is known at all times, the world can be modeled as a Markov Decision Process (MDP). MDPs have been studied extensively and many methods are known for determining optimal courses of action, or policies. The more realistic case where state information is only partially observable, Partially Observable Markov Decision Processes (POMDPs), have received much less attention. The best exact algorithms for these problems can be very inefficient in both space and time. We introduce Smooth Partially Observable Value Approximation (SPOVA), a new approximation method that can quickly yield good approximations which can improve over time. This method can be combined with reinforcement learning methods, a combination that was very effective in our test cases.

1 Introduction

Markov Decision Processes (MDPs) have proven to be useful abstractions for a variety of problems. When a domain fits into the MDP framework, a variety of methods can be used that are practical for small- to medium-sized problems. Unfortunately, many interesting domains cannot be modeled as MDPs. In particular, domains in which the state of the problem is not fully observable at all times cannot be modeled as MDPs. Partially Observable Markov Decision Processes (POMDPs) extend the MDP framework to include partially observable state information. With this extension, we are able to model a larger and more interesting class of problems, but we are no longer able to use the solution methods that exist for MDPs.

POMDP algorithms are much more computationally intensive than their MDP counterparts. The reason for this complexity is that uncertainty about the true state of model induces a probability distribution over the model states. Most MDP algorithms work by determining the value of being in one of a finite number of discrete states, while most POMDP algorithms are forced to deal with probability distributions. This difference changes a discrete optimization problem into a problem that is defined over a continuous space. This increase in complexity is manifested in the performance of POMDP algorithms: The best algorithms can take prohibitively large amounts of time even for very small problems.

Our approach, Smooth Partially Observable Value Approximation (SPOVA), uses a smooth function that can be adjusted with gradient descent methods. This provides an extremely simple improvement rule that is amenable to reinforcement learning methods and will permit an agent to gradually improve its performance over time.

In our test cases, we found that agents using this rule could rapidly improve their behavior to near-optimal levels in a fraction of the time required to run traditional POMDP algorithms to completion.

The following section will introduce the MDP formalism, and section 3 will show how this can be extended to include partial observability. Section 4 introduces a smooth approximation to the max function that is the basis of our SPOVA algorithms. A simple gradient descent SPOVA algorithm is described in section 5, and results for this algorithm are presented in section 6, where it finds optimal policies for two test worlds. An approach based on simulated exploration and reinforcement learning is introduced in section 7, where results are presented showing this method rapidly finds good policies. Section 8 briefly discusses other related work, and section 9 contains concluding remarks.

2 Markov Decision Processes

One useful abstraction for modeling uncertain domains is the Markov Decision Process or MDP. An MDP divides the world into states with actions that determine transition probabilities between these states. The states are chosen so that each state summarizes all that is known about the current status of the world: the probability of the next state is a function of the current state and action only, not any of the previous states or actions. More formally, we say that for any action a and string of states and actions $S_1a_1 \ldots S_ta_t$, $P(S_{t+1}|a_tS_ta_{t-1}\ldots a_1S_1) = P(S_{t+1}|a_tS_t)$. This is called the Markov Property.

An MDP is a 4-tuple, (S, A, T, R) where S is a finite set of states, A is a finite set of actions, T is a mapping from $S \times A$ into distributions over the states in S, and R is a reward function that maps from S to real-valued rewards. This paradigm can be extended to distributions over rewards, or to map from $S \times A$ to rewards or distributions over rewards. There may be an additional element, I, which specifies an initial state.

A policy for an MDP is a mapping from S to actions in A. It can be an explicit mapping, or it can be implicit in a value function, V, that maps from elements of S to real values. This value function represents the value of being in any state as the expected sum of rewards that can be garnered from that point forward. We can use a value function to assign actions to states in S by choosing the action that maximizes the expected value of the succeeding states. Policies can be defined for two types of problems, finite-horizon, where the number of steps or actions permitted has a hard limit, and infinite-horizon, where there is no fixed time limit. The infinite-horizon case still can respect the value of time by incorporating a cost or negative reward with each step, or by discounting future rewards by a discount factor, $0 \leq \beta \leq 1$.

For any MDP, there exists an optimal value function V^* that can be used to induce an optimal policy. The present value of the rewards expected by an agent acting on an optimal policy will be at least as great as that received by an agent under any other policy. There are several methods for determining optimal policies for MDPs. One effective method for determining a value function for the infinite horizon case is value iteration [Bellman, 1957]. If the transition probabilities for the model are not known, reinforcement learning [Sutton, 1988] can be used to learn an optimal policy through exploration.

When separate value functions are maintained for each action, these functions are often called Q-functions. When reinforcement learning is used to learn Q-functions it is called Q-learning [Watkins, 1989]. Our algorithms do *not* maintain separate value functions for each action. As we will discuss below, we regard this as simply an implementation detail and not an important distinction for our approach.

3 Partial Observability

It is important to realize that although actions have uncertain outcomes in MDPs, there is never any uncertainty about the current state of the world. Before taking any action, an agent may be uncertain about the consequences of its action, but once the action is taken, the agent will know the outcome. This can be an extremely unrealistic assumption about the ability of an agent's sensors to distinguish between world states.

A Partially Observable Markov Decision Process (POMDP) is just like an MDP with outputs attached to the states. The outputs can be thought of as sensor observations that provide (usually) uncertain information about the state of the world. as hints about the true state of the world, or as sensor inputs. More formally, a POMDP is a 5-tuple (S, A, T, R, O), where S, A, T, and R are defined as in an MDP and O maps from states in S to a set of outputs. Note that if O assigns a unique output to every state and the initial state is known, then the POMDP becomes an MDP because the state information is fully observable. POMDPs can be extended to make S map from states to distributions over outputs, or from $S \times A$ to outputs or distributions over outputs. There may be an additional element, I, that determines an initial distribution over states.

The change to partial observability forces an important change in the type of information an agent acting in a world must maintain. For the fully observable case, an agent will always know what state it is in, but for the partially observable case, an agent that wishes to act optimally must maintain considerably more information. One possibility is a complete history of all actions taken and all observations made. Since this representation can become arbitrarily large, the maintenance of a joint probability distribution over the states in S often is more tractable. This distribution sometimes is referred to as a belief state.

It can be shown that the Markov property holds for the belief states induced by a POMDP. This means that in principle, we can construct an MDP from the belief states of a POMDP, find an optimal policy for the MDP and then use this policy for the POMDP. Unfortunately, most interesting POMDPs induce a very large or infinite number of belief states, making direct application of MPD algorithms to POMDPs impractical.

A survey of existing POMDP algorithms [Lovejoy, 1991] shows that many POMDP algorithms work by constructing a finite representation of a value function over belief states, then iteratively updating this representation, expanding the horizon of the policy it implies, until a desired depth is reached. For some classes of problems [Sondik, 1971] infinite-horizon policies will have finite representations and value functions can be obtained for these problems by expanding the horizon until the value function converges to a stable value. In practice, infinite-horizon policies often can be approximated by extremely long finite horizons even if convergence is not obtained. Regardless of whether they are run to convergence, existing exact algorithms can take an exponential amount of space and time to compute a policy, even if the policy itself does not require an exponential size representation. These drawbacks have led to a number of approximation algorithms that work by discretizing the belief space. The most advanced methods dynamically adjust the resolution of the discretization for different parts of the belief space, but it is unclear whether this can be done efficiently for large problems.

It is worth noting for the reader unfamiliar with this area that most POMDPs with known solutions have less than 10 states and that exact solutions to POMDPs with tens of states can take anywhere from minutes to days if convergence is obtained at all.

We will introduce a new approximate method for determining infinite horizon policies for POMDPs. This method differs from existing methods in that it uses a continuous and differentiable representation of the value function.

4 The Differentiable Approximation

The first, and perhaps most important, decision that must be made in any approach to this problem is how to represent the value function. Sondik showed [1971] that an optimal finite-horizon value function can be represented as the max over a finite set of linear functions of the belief state. For a belief state b, a vector representing a distribution over the states of the world, the value function can be represented as:

$$V(b) = \max_{\gamma \in \Gamma} b \cdot \gamma$$

where Γ is a set of vectors the same dimension as b, defining planes in value $\times$ belief space. Each γ_i in Γ can be shown to represent a fixed policy, meaning that we are maximizing over a set of policies to find the one that is best in a particular region of the belief space. (See [Littman, 1994] for an in-depth interpretation of the γ vectors.) Graphically, γ_i is a hyperplane in value space and the max of these functions forms a convex piecewise-linear surface. The significance of Sondik's result is that it provides a potentially compact means of representing the optimal value function for finite-horizon problems, although it does not make any guarantees that $|\Gamma|$ will be tractably small.

For very large horizons, the value function may be quite smooth as it may be comprised of a very large number of vectors. For infinite horizons, the value function may be comprised of an infinite number of pieces, which means that it is likely to be smooth in at least parts of the belief space. In any case, because it is the maximum of a set of linear functions, it will be convex. For these reasons, a good candidate for a differentiable approximation of the infinite horizon

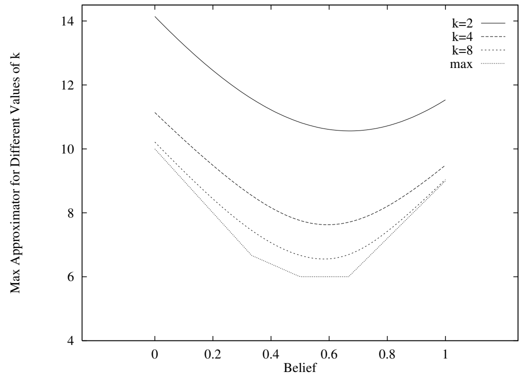

Figure 1: Closeness of the *MAX* approximation as k increases.

value function would be a convex function that behaves like max. The following function works rather nicely and is the foundation of the smooth approximation made by SPOVA: [1]

$$V(b) = \sqrt[k]{\sum_{\gamma \in \Gamma} (b \cdot \gamma)^k}.$$

To keep things simple, we will assume that the true value function, V^*, is always positive and that the individual components of the γ_is, the γ_{i_j}s, are all positive. This assumption comes with no loss of generality since we easily can shift the function into the positive part of the value space to satisfy these conditions. This can be done by replacing $(b \cdot \gamma)$ with $([b \cdot \gamma] + w)$ where w is a constant offset.

Since the γ_{i_j}s are always positive, the second partial derivative in each of the dimensions is always positive, and the function is always convex. The function will behave like an over-estimate of max that is smoothed at the corners. Figure 1 shows a two-dimensional example of how this works. We have chosen $\Gamma = \{(9,0),(4,8),(6,6),(0,10)\}$ and graphed our differentiable max approximator for different values of k. The convex piecewise-linear function below the smooth curves is the max function. (Only one belief dimension is shown because the second is 1 minus the first.) Notice that as k increases, the approximation approaches the shape of the convex surface that is the max of the linear functions. The height of the function is less important than the shape here since the policy induced by a value function depends on relative, not absolute, value assignments.

The k parameter gives us a great deal of flexibility. For example, if we believe that the infinite horizon value function can be represented by the max of a small set of linear functions, we may choose a large value for k and try for a very close approximation. On the other hand, if we believe the optimal infinite horizon value function is complex and highly textured, requiring more components than we have time or space to represent, a smaller value of k will smooth the approximate value function to partially compensate for a lower number of γ vectors.

5 The basic SPOVA algorithm

The main advantage of a continuous representation of the value function is that we can use gradient descent to adjust the parameters of the function to improve our approximation. Ideally, we could use data points from the optimal value function, V^*, to construct Γ. Such information generally is not available, but an approach similar to value iteration for MDPs can be to make our value equation look more like V^*. We know from value iteration that the optimal value equation for an MDP must satisfy the following constraint:

$$\forall s : V^*(s) = R(s) + \beta \max_{a \in A} \sum_{s' \in S} P(s'|s,a) V^*(s')$$

Since a POMDP induces an MDP in the belief states of the POMDP, we know that this equation must hold for the optimal value function for POMDPs as well. This gives us a strategy for improving the value function: Search for inconsistencies in our value function, then adjust the parameters in the direction that minimizes these inconsistencies. This is done by computing the Bellman residual [Bellman, 1957],

$$E(b) = V(b) - (R(b) + \beta \max_{a \in A} \sum_{b' \in next(b,a)} P(b'|b,a) V(b'))$$

where $next(b,a)$ is the set of belief states reachable from b on taking action a. We can then adjust the γs in the direction that minimizes the error. By using a smooth max approximation described above, we are able to use a typical gradient descent approach: $\Delta w = \alpha E(b) \nabla_w V(b)$, where α is interpreted as a step size or learning rate. In this case, the weights correspond to the components of the γ vectors, so the update for the j^{th} component of the i^{th} γ vector, γ_{i_j}, turns out to be:

$$\Delta \gamma_{i_j} = \frac{\alpha E(b) b_j (b \cdot \gamma_i)^k}{V(b)^k}$$

This equation for the gradient has several appealing properties. The $(b \cdot \gamma_i)^k$ part increases with the contribution γ_i makes to the value function so the γ_is that contribute most to the value function are changed the most. This is then multiplied by b_j, reflecting the influence of the probability of being in state j on the j^{th} component of the gradient of γ_i. We also can interpret k as a measure of how "rigid" the system is. For small values of k, many weights will be updated with each change. However, for large values of k, the $(b \cdot \gamma_i)^k$ component of the gradient will permit only minuscule changes to all but the the γ_i that maximize $b \cdot \gamma_i$. Figure 2 shows the SPOVA algorithm.

For each belief state b

$E \leftarrow V(b) - (R(b) + \beta \max_{a \in A} \sum_{b' \in next(b,a)} P(b'|b,a) V(b'))$

For i from 1 to $|\Gamma|$

For j from 1 to n

$\gamma_{i_j} \leftarrow \gamma_{i_j} + \alpha E b_j (\gamma_i \cdot b)^k / V(b)^k$

Figure 2: The SPOVA algorithm.

Since it is impossible to sample all possible belief states, we used the simple approach of randomly selecting belief states. Empirically, we found that we obtained the best results when we varied k during the run-time. Typically, we would start k at 1.2 and increase k linearly until it reached 8.0 when 75% of the requested number of updates were performed. As shown in Figure 1, small values of k make smoother and more general approximations. Small values of k also spread the effect of updates over a wider area, in some sense increasing

[1] There are other possible choices for "soft" max approximations. See, for example, [Martinetz *et al.*, 1993].

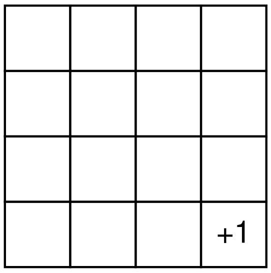

Figure 3: The 4x4 World.

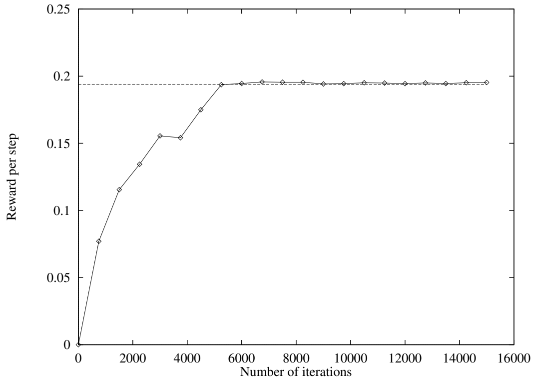

Figure 4: Policy quality vs. number of iterations for gradient descent in the 4×4 world.

the "energy" of the system. This gradual increase in k can be thought of as a form of simulated annealing.

The updates can be repeated until some termination condition is met, either a fixed limit in the number of iterations, a minimum number of consecutive samples processed without E exceeding some threshold, or perhaps something more problem-specific.

While we do not yet have a convergence proof for this algorithm, we are optimistic that with enough iterations, decaying α, and sufficiently large Γ that as k tends toward infinity, the value function will converge to the optimal value function if the function has a finite piecewise-linear representation. This is because our error function will become arbitrarily close to the Bellman residual as k increases. For a large number of updates, the system should move towards its only stable equilibrium point, the point at which the value function is consistent and, therefore, optimal for all points in the belief space.

One question that has not been addressed is how to pick the number of γ vectors to use. For a sub-optimal number of vectors, the gradient descent approach will adjust these vectors in the direction of lower error even though convergence may not be possible. Our algorithms do not yet automatically determine the optimal number of vectors needed to converge to the value function in the limit. One practical way to incorporate this ability would be to code what we did by hand: use a binary search to find the smallest number of vectors that gives an optimal policy (one that is no worse than the best policy produced with a larger number of vectors).

6 SPOVA results

We tried the basic SPOVA algorithm initialized with random γ vectors for two grid worlds that have appeared in the literature. The first, shown in Figure 3, is a 4×4 world from [Cassandra *et al.*, 1994]. Movement into adjacent squares is permitted in the four compass directions but an attempt to move off the edge of the world has no effect, returning the agent to its original state with no indication that anything unusual has happened. All states have zero reward and the same appearance, except for the bottom right state, which has a +1 reward and a distinctive appearance.

The initial state for this problem is a uniform distribution over all but the bottom right state. Any action taken from the bottom right state results in a transition to any one of the remaining zero reward states with equal probability (i.e., return to the initial distribution). For this problem we are interested in the optimal infinite-horizon policy with a discount factor of β = 0.8. With a moment's thought, it should be clear that the optimal policy for this model alternates between moving East and South. This does not mean that the optimal infinite horizon value function is easily obtainable. In fact, there are 887 belief states that are reachable from the initial state and the optimal value function defined over all belief states requires 20 vectors using Sondik's representation.

We ran gradient descent with just 1 vector for 50,000 iterations and compared the value of the resulting approximate policy to the value of the optimal policy at 1000-iteration intervals. We did this by taking a snapshot of the value function at each interval, then simulating 10,000 steps through the world and counting the average reward per step garnered during this period. This provides an estimate of the current policy quality. We compared this against the policy quality for the known optimal policy for the same time period. Figure 4 shows a graph of the average reward garnered per step vs. the number of iterations performed. The horizontal line is the value of the optimal policy, computed using the Witness algorithm [Cassandra *et al.*, 1994], perhaps the fastest known exact algorithm. Both algorithms required time on the order of CPU minutes.

Our second problem, shown in Figure 5, is from [Russell and Norvig, 1994]. It is a 4×3 grid-world with an obstruction at (2,2). The coordinates are labeled in x,y pairs, making (1,3) the top left. There is no discounting, but a penalty of 0.04 is charged for every step that is taken in this world. The two reward states, +1 and -1, are both directly connected to a single zero-reward absorbing state. Originally, this problem was used in a fully observable context, but we have made it partially observable by limiting state information to that obtained from one east-looking and one west-looking wall detector. Each is activated when there is a wall in the immediately adjacent square. For example, this makes (1,1), (1,3), and (3,2) indistinguishable. The initial state is selected uniformly at random from the nonterminal states.

Unlike the 4×4 world, transitions are not deterministic. Every action succeeds with probability 0.8 and fails with probability 0.2 moving the agent in a direction perpendicular from the intended one. If such a movement is obstructed by a wall, then the agent will stay put instead. Moving right from (1,3), for example, will move the agent right with probability 0.8, down with probability 0.1 and nowhere with probability 0.1.

We ran the gradient descent method for $400,000$ iterations with 3 vectors and obtained the results in Figure 6. The algorithm requires many samples, about 250,000 (42 CPU minutes), before it has enough data in the relevant portion of the space to calculate an approximately optimal policy. The comparison policy shown in the figure with a reward per step of 0.1108 was obtained after over 12 CPU hours using

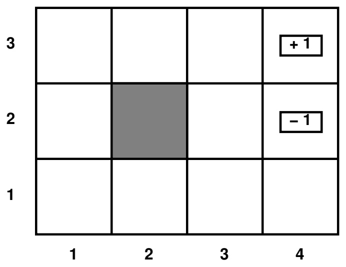

Figure 5: The 4×3 world from [Russell and Norvig, 1994].

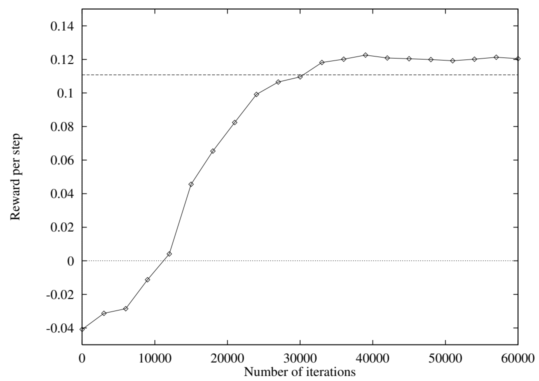

Figure 6: Performance of the gradient descent algorithm on the 4×3 world, showing the policy quality as a function of the number of iterations.

the Witness algorithm and uses 30 vectors. In this case the Witness algorithm did not converge, although recent results in [Littman *et al.*, in press] indicate that convergence or near convergence may not be necessary in all cases to obtain a good policy from the Witness Algorithm.

One perhaps surprising aspect of our approximation method is that the number of vectors required is drastically lower than that for an exact solution. We were initially surprised to discover that the 4×4 problem requires a single vector, making the value function linear. Part of the savings comes from the fact that our simulations considered only reachable belief states, while exact solutions like the Witness algorithm construct policies that cover the entire belief space. Also many more vectors may be required to specify a correct value function than are needed to specify a correct policy. From the policy perspective, it is sufficient to know the relative value of all of the belief states, not their exact value, making the shape of the value function much more important than the specific values it returns. For the 4×4 problem, any function that assigns a higher value to belief states that suggest that the agent is closer to the southeast corner of the world will be sufficient. A simple linear function is all that is needed here.

The use of a smooth function also can reduce the number of vectors required. For example, a complex bend that is formed by many hyperplanes in the exact value function often can be approximated very closely by a single smooth bend.

7 A reinforcement learning approach

The straightforward gradient descent method can bring our approximate value function fairly close to the exact one. With a sufficient number of iterations, the average difference over the entire state space will be very small. A possible shortcoming of this method is that it cannot guarantee that the approximation will not differ significantly from V^* at critical parts of the space, such as the initial state. In addition, random selection of belief states may waste time refining the value function in parts of the belief space that would rarely, if ever, be visited by an agent following an optimal policy. Finally, the gradient descent method, like some exact methods, does not make use of information about the initial distribution over states. This information can greatly limit the number of reachable belief states, making the problem easier.

We have implemented a second variation on our SPOVA approach, SPOVA-RL (Smooth Partially Observable Value Approximation with Reinforcement Learning), which avoids these problems. The algorithm uses the known model to simulate transitions in the environment. Effectively, it "explores" the belief state space with the aim of finding high-utility regions. This tends to focus the updates to the value function on belief states that are likely to be encountered by an agent using an optimal or near-optimal policy. The SPOVA-RL update rule for a belief state b is show in Figure 7.

$a \leftarrow$ best action according to V
$b' \leftarrow$ simulated result of taking a in b.
$E_{RL}(b) \leftarrow V(b) - (R(b) + V(b'))$
For i from 1 to $|\Gamma|$
    For j from 1 to n
        $\gamma_{i_j} \leftarrow \gamma_{i_j} + \alpha E_{RL}(b) b_j (\gamma_i \cdot b)^k / V(b)^k$

Figure 7: The SPOVA-RL update rule.

For each transition, the algorithm applies the same γ_{i_j} update as the gradient descent algorithm, but we compute E_{RL} with to respect the belief state that is encountered in the simulation, rather than by maximizing over all possible successor states. Where b is the belief state at time t and b' is the belief state at time $t + 1$ we compute:

$$E_{RL}(b) = V(b) - (R(b) + V(b'))$$

To ensure sufficient exploration of the world, we chose initial values for the γ vectors that guaranteed an overestimate for every possible belief state. This forced the algorithm to disprove the optimistic estimates by visiting different areas of the belief space. This rather simplistic policy was sufficient for our examples, but we are investigating the application of some of the methods that have been used for MDPs to improve the speed of convergence and to provide stronger guarantees that enough of the belief space will be covered.

We ran the algorithm on the same two worlds as before. The results are shown in Figures 8 and 9. SPOVA-RL finds an approximately optimal policy for the 4×4 world in about 80 iterations (1.4 seconds), and for the 4×3 world in about 6000 iterations (59 seconds).

By focusing its efforts on the most important states in the belief space, SPOVA-RL is able to learn a nearly optimal policy extraordinarily quickly. While some of this speed may come at the expense of accurate value estimations for rarely visited states, this is an acceptable price to pay for many domains.

As a final experiment, we investigated the world shown in Figure 10. This world is designed to require a value function with more than one vector. (Intuitively, being in a linear combination of the A-states is much worse than being definitely in one or the other.) Figure 11 shows the expected result, namely that SPOVA-RL effectively approximates an optimal 3-vector policy.

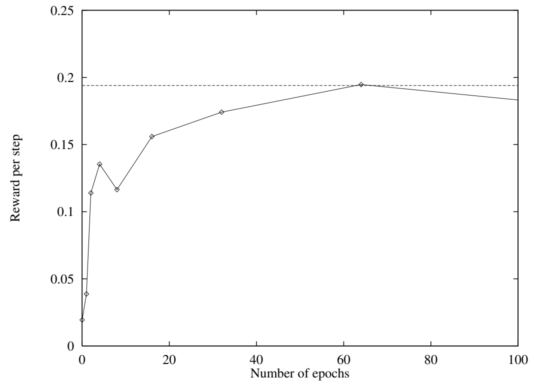

Figure 8: Performance of the SPOVA-RL algorithm on the 4×4 world, showing the policy quality as a function of the number of epochs.

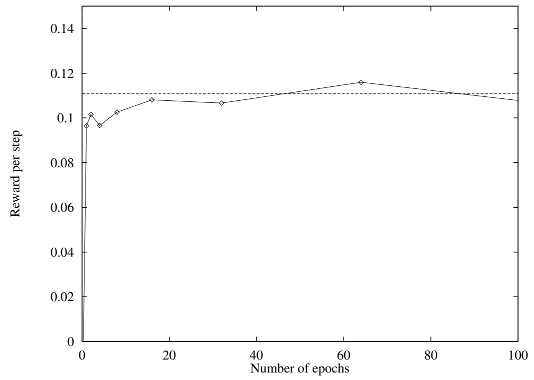

Figure 9: Performance of the SPOVA-RL algorithm on the 4×3 world, showing the policy quality as a function of the number of epochs.

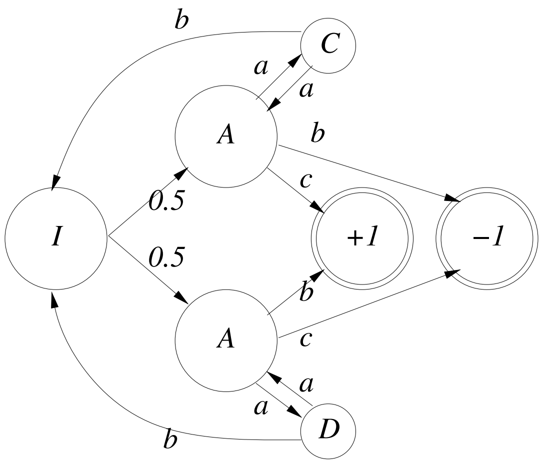

Figure 10: A simple domain requiring more a nonlinear value function. States labelled *A* are indistinguishable, but actions *b* and *c* can lead either to a +1 or a –1 reward depending on which if the *A*-states the agent is in. Action *a* leads to a distinctive state (either *C* or *D*), which enables the agent to find out where it is.

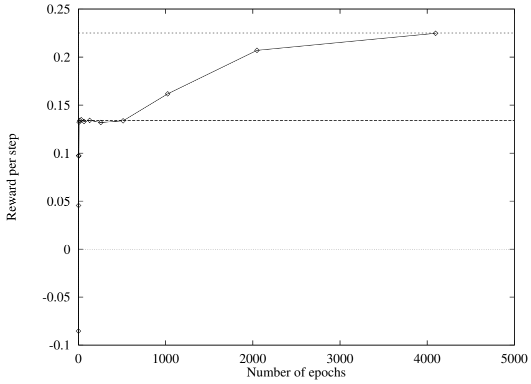

Figure 11: Performance of the SPOVA-RL algorithm on the environment shown in Figure 10. With one vector, SPOVA-RL finds a policy of value 0.134 (the lower horizontal line). With three vectors, SPOVA-RL quickly finds the optimal one-vector policy, but after about 600 iterations abandons it in favour of the more complex 3-vector policy, which eventually reaches the optimal value of 0.225 (the upper horizontal line).

8 Relation to other work

Many MDP and POMDP algorithms determine Q values rather than a single value function as we have done here. The problem of determining the best action from an ordinary value function requires an agent to consult a model to simulate one step into the future and consider the value of possible next states. An agent using Q values does not need to look ahead in this fashion since the value of each action is represented directly. In the case of Q learning, a model is not even needed to construct the Q values as they are learned directly from agent's experience. This so-called "model-free" property of Q-learning does not carry over to POMDPs. The agent must know something about the dynamics of the world if a compact state description is to be maintained over time. Without a model, this state description cannot be evolved and an agent would be forced either to guess about its true location or to define value functions or Q functions over its entire history. Thus, reengineering a POMDP algorithm to compute Q functions rather than a value function may change the analysis of the algorithm, but it does not change fundamentally the nature of the problem as it is alleged to do for MDPs. In fact for the SPOVA implementations we have discussed here, it is a trivial change.

Another approach to the problem of partial observability is to simply pretend that the sensor observations correspond exactly to states. *Deterministic* policies constructed for this "sensor space" usually fail miserably, typically resulting in looping behavior. This can be alleviated to some extent by using *randomized* policies of the kind first proposed for use in games of partial information. Jaakola *et al.* [in press] have shown how to learn from reinforcement using randomized policies, demonstrating that the approach is not unreasonable in some cases.

A linear value approximator is combined with a clever model learning mechanism in [McCallum, 1993] and [Chrisman, 1992]. It may be possible to generalize their approach to include more complex functions like those represented by SPOVA. A neural network based approach is used in [Lin and Mitchell, 1992]. They consider a variety of approaches that can make use of an agent's history to learn hidden state information. The idea of a smoothed or "soft" max has been around for a while. It is the basic idea behind the use of the Boltzman distribution for action selection in [Watkins, 1989] and a similar approach has been used in neural networks in, for example, [Martinetz *et al.*, 1993]. We suspect that it may be possible to adapt these approximators for use in POMDPs using a similar approach to the one described here although we have not yet investigated this fully. In recent work by Littman et al. [in press] an update rule was developed independently that can be interpreted as a special case of SPOVA. This was shown to be adequate for determining good policies for problems with over 30 states.

9 Conclusions and future work

We have investigated SPOVA, an approximation scheme for partially observable Markov decision problems based on a continuous, differentiable representation of the value function. A simple "value iteration" algorithm using gradient descent and random sampling is shown to find approximately optimal policies but requires a large number of samples from the belief state space. We conjectured that many of these samples correspond to very unlikely or even unreachable belief states, and therefore designed SPOVA-RL, a reinforcement learning algorithm that focuses its value function updates on belief states encountered during actual exploration of the state space. SPOVA-RL was able to solve the 4×4 and 4×3 worlds very quickly, suggesting that optimism concerning the value of generalized approximation methods for POMDPs may be justified.

The next steps are to tackle larger problems, to obtain convergence results, and to incorporate methods for learning the environment model. We currently are investigating the application of a new algorithm for learning dynamic probabilistic networks (DPNs) [Russell *et al.*, 1994]. Such algorithms can find decomposed representations of the environment model that should allow very large state spaces to be handled. Furthermore, the DPN provides a reduced representation of the belief state that may facilitate additional generalization in the representation of the value function. We plan to use the overall approach to learn to drive an automobile.

10 Acknowledgement

Members of the U.C. Berkeley MDP reading group, especially Daphne Koller, provided helpful suggestions and feedback on the ideas contained in this paper. We are very grateful to Tony Cassandra and Michael Littman for sharing their Witness algorithm results with us and running their algorithm on our 4×3 world. Michael Littman also provided extensive comments on an early draft of this paper. Tim Huang provided help with formatting and figures.

References

[Bellman, 1957] Richard Ernest Bellman. *Dynamic Programming*. Princeton University Press, Princeton, New Jersey, 1957.

[Cassandra *et al.*, 1994] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 1023–1028, Seattle, Washington, August 1994. AAAI Press.

[Chrisman, 1992] Lonnie Chrisman. Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 183–188, San Jose, California, July 1992. AAAI Press.

[Jaakola *et al.*, in press] Tommi Jaakola, Satinder P. Singh, and Michael I. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. In *Neural Information Processing Systems 7, to appear*, in press.

[Lin and Mitchell, 1992] Long-Ji Lin and Tom M. Mitchell. Memory approaches to reinforcement learning in non-Markovian domains. Technical report, Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1992.

[Littman *et al.*, in press] Michael L. Littman, Anthony R. Cassandra, and Leslie P. Kaelbling. Learning policies for partially observable environments: Scaling up. In *Proceedings of the Twelfth International Conference on Machine Learning, to appear*, in press.

[Littman, 1994] Michael L. Littman. The witness algorithm: Solving partially observable Markov decision processes. Technical report, Computer Science Department, Brown University, Providence, Rhode Island, 1994.

[Lovejoy, 1991] W. S. Lovejoy. A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, 28(1–4):47–66, April 1991.

[Martinetz *et al.*, 1993] Thomas M. Martinetz, Stanislav G. Berkovich, and Klaus J. Schulten. "neural-gas" network for vector quantization and its application to time series prediction. *IEEE Transactions on Neural Networks*, SSC-4:558–569, 1993.

[McCallum, 1993] Andrew R. McCallum. Overcoming incomplete perception with utile distinction memory. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 190–196, Amherst, Massachusetts, July 1993. Morgan Kaufmann.

[Russell and Norvig, 1994] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, New Jersey, 1994.

[Russell *et al.*, 1994] Stuart J. Russell, John Binder, and Daphne Koller. Adaptive probabilistic networks. Technical Report UCB/CSD-94-824, Computer Science Division, University of California at Berkeley, July 24 1994.

[Sondik, 1971] E. J. Sondik. *The Optimal Control of Partially Observable Markov Decision Processes*. PhD thesis, Stanford University, Stanford, California, 1971.

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

[Watkins, 1989] C. J. Watkins. *Models of Delayed Reinforcement Learning*. PhD thesis, Psychology Department, Cambridge University, Cambridge, United Kingdom, 1989.