BI-POMDP: Bounded, Incremental Partially-Observable Markov-Model Planning

Richard Washington

Caelum Research Corporation
NASA Ames Research Center
MS 269-1
Moffett Field, CA 94035-1000
rwashington@mail.arc.nasa.gov

Abstract. Given the problem of planning actions for situations with uncertainty about the action outcomes, Markov models can effectively model this uncertainty and offer optimal actions. When the information about the world state is itself uncertain, partially observable Markov models are an appropriate extension to the basic Markov model. However, finding optimal actions for partially observable Markov models is a computationally difficult problem that in practice borders on intractability. Approximate or heuristic approaches, on the other hand, lose any guarantee of optimality or even any indication of how far from optimal they might be.
In this paper, we present an incremental, search-based approximation for partially observable Markov models. The search is based on an incremental AND-OR search, using heuristic functions based on the underlying Markov model, which is more easily solved. In addition, the search provides a bound on the possible error of the approximation. We illustrate the method with results on problems taken from the related literature.

1 Introduction

Uncertainty abounds in real-world domains. Given the unpredictability of processes that act in the world and the complexity of agents and objects in the world, models of such domains necessarily rest uncertain. The uncertainty pervades models of actions as well as models of the state of the world.

Consider the problem of medical diagnosis and therapy. The state of the patient is only known indirectly through tests performed on the patient and sensors attached to the patient, from which one can induce in which states the patient might be. Based on this information, a therapeutic course of action is chosen that best addresses the possible problems, but the effects of this action on the patient are only imperfectly known. So the therapy must allow for multiple possible outcomes.

Markov models [1] are useful for representing and reasoning about uncertain domains. The process, whether it be a patient, robot, or chemical plant, is represented as a set of discrete states. At any moment in time, the process is in one

of the states. When an action is taken, the state of the process changes, with the resulting state dependent on the action and predefined probabilities of transitions between states. The basic Markov model represents the uncertainty about action outcomes using the transition probabilities and offers optimal actions in the face of this uncertainty.

The basic Markov model can be extended to handle the case of partial observability, where the state of the process is not necessarily known. Instead, there are observations available that provide information about the state but may not fully identify it. So the choice of the optimal action must be made knowing only a probability distribution over states rather than the true state. This makes the problem of determining the optimal action much more difficult than for the fully observable case, and in fact is currently only possible for very small problems [4].

Approximations for partially-observable Markov models exist, but they tend to fall prey to a few problems. Either they themselves do not completely avoid the complexity problems [5], or they ignore aspects of the model that are necessary to ensure optimal actions [9,8].

In this paper, we present an incremental approach for partially observable Markov models that in the limit achieves the optimal action, and in the meantime provides an approximate solution that can be used when time is of the essence. In addition, the partial solution comes with a provable bound on its error (a best- and worst-case estimate), so that the underlying decision-making system can see the possible impact of taking the action at each moment. For example, in the medical example this would correspond to having an estimate of the optimistic and pessimistic estimates of the morbidity of a therapy (if that were the measure of utility), thus avoiding the problem of taking a therapy that looks good, only to find out that it in fact has a disastrous outcome in reality.

The organization of the paper is as follows. We present Markov models and the partially observable version thereof. Then we describe how to reason about partially observable Markov models as a search problem. Then we describe how this search can be performed incrementally, providing both optimistic and pessimistic bounds, which in turn can direct the search process. We present results on problems from the literature to illustrate the method and its effectiveness. Finally, we discuss the limitations of the method and current directions for research.

2 Partially Observable Markov Decision Processes

In this section we briefly review Markov processes, and in particular POMDPs. We will borrow the notation of [6], altering it only as required for the problem at hand; the reader can refer there for a more complete explanation of the framework.

We assume that the underlying process, the *core process*, is described by a finite-state, stationary Markov chain. The core process is captured by the following information:

- a finite set $\mathcal{N} \equiv \{1, \ldots, N\}$, representing the possible states of the process
- a variable $X_t \in \mathcal{N}$ representing the state of the core process at time t
- a finite set $\mathcal{A}$ of actions available
- a matrix $P = [p_{ij}], i, j \in \mathcal{N}$ specifying transition probabilities of the core process: $P(a) = [p_{ij}(a)]$ specifies the transition probabilities when action $a \in \mathcal{A}$ is chosen
- a reward matrix $R = [r_{ij}], i, j \in \mathcal{N}$ specifying the immediate rewards of the core process: $R(a) = [r_{ij}(a)]$ specifies the reward received when the action $a \in \mathcal{A}$ is executed. We will use the shorthand

$$\varrho_i(a) = \sum_{j \in \mathcal{N}} r_{ij}(a) p_{ij}(a)$$

to denote the reward of taking action a when in state i, and

$$\varrho(a) = \{\varrho_1(a), \ldots, \varrho_N(a)\}.$$

So at time t, the core process is in state $X_t = i$, and if an action $a \in \mathcal{A}$ is taken, the core process transitions to state $X_{t+1} = j$ with probability $p_{ij}(a)$, receiving immediate reward $r_{ij}(a)$.

Actions are chosen by a policy that maps states to actions. The optimal policy is the policy that maximizes the utility of each state. The value of a state under the optimal policy (given full observability) is defined as:

$$v(i) = \max_{a \in \mathcal{A}} \left\{ \varrho_i(a) + \beta \sum_{k \in \mathcal{N}} v(k) p_{ik}(a) \right\}$$

where $0 \leq \beta < 1$ is a discount factor (this ensures a bounded value function).

However, in a partially observable MDP, the progress of the core process is not known, but can only be inferred through a finite set of observations. The observations are captured with the following information:

- a finite set $\mathcal{M} \equiv \{1, \ldots, M\}$ representing the possible observations
- a variable $Y_t \in \mathcal{M}$ representing the observation at time t
- a matrix $Q = [q_{ij}], i \in \mathcal{N}, j \in \mathcal{M}$ specifying the probability of seeing observations in given states: $Q(a) = [q_{ij}(a)]$, where $q_{ij}(a)$ denotes the probability of observing j from state i when action $a \in \mathcal{A}$ has been taken
- a state distribution variable $\pi(t) = \{\pi_1(t), \ldots, \pi_N(t)\}$, where $\pi_i(t)$ is the probability of $X_t = i$ given the information about actions and observations
- an initial state distribution $\pi(0)$.

At time t, the observation of the core process will be Y_t. If action $a \in \mathcal{A}$ is taken, we can define a function to determine Y_{t+1}. In particular, we define

$$\gamma(j|\pi(t), a) = \sum_{i \in \mathcal{N}} q_{ij}(a) \sum_{k \in \mathcal{N}} p_{ki}(a) \pi_k(t) \tag{1}$$

as the probability that $Y_{t+1} = j$ given that action $a \in \mathcal{A}$ is taken at time t and the state distribution at that time is $\pi(t)$.

To determine the state distribution variable $\pi(t+1)$, we define the transformation T as follows:

$$\begin{aligned}\pi(t+1) &= T(\pi(t)|j,a) \\ &= \{T_1(\pi(t)|j,a), \ldots, T_N(\pi(t)|j,a)\}\end{aligned}$$

where

$$T_i(\pi(t)|j,a) = \frac{q_{ij}(a)\sum_{k\in\mathcal{N}} p_{ki}(a)\pi_k(t)}{\sum_{l\in\mathcal{N}} q_{lj}(a)\sum_{k\in\mathcal{N}} p_{kl}(a)\pi_k(t)}, \tag{2}$$

for $i \in \mathcal{N}$, and where $\pi(t)$ is the state distribution at time t, $a \in \mathcal{A}$ is the action taken at that time, resulting in observation $j \in \mathcal{M}$.

Actions are chosen by a decision rule (or plan) that maps state distributions to actions. The utility of a state distribution π under the optimal decision rule can be computed by the value function:

$$V_P(\pi) = \max_{a\in\mathcal{A}} \left\{ \pi \cdot \varrho(a) + \beta \sum_{j\in\mathcal{M}} V_P[T(\pi|j,a)]\gamma(j|\pi,a) \right\}. \tag{3}$$

3 The problem with MDP-based solutions

It is better to make intelligent use of the time available rather than doing something that is always a bit wrong. This is one of the fundamental premises underlying the work presented here. The goal is to have something that converges towards the optimal solution. The simple use of an MDP-based solution method, albeit fast, is guaranteed not to work in certain cases.

To illustrate the importance of solving the POMDP rather than simply the underlying MDP, consider the problem shown in Figure 1. The agent starts in the center square, but without knowing which of 4 orientations it is in. The agent can only see the walls around it, and the walls have distinctive markings, so that once it is in a side area, it will know where it is. If we use a weighted MDP-based policy or some variant thereof [9], the action recommended will be to turn forever, since there will be two possible states with a recommendation to turn one direction, one possible state with a recommendation to turn the other direction, and one possible state with a recommendation to advance. In fact, the optimal action is to advance, read the wall if it isn't the goal, and then head toward the goal. This problem generalizes to any large open space.

4 Search-based POMDP Approximation

For an initial state distribution, there is a set of actions that may be taken. Each of these actions leads to a set of possible observations, each weighted by the probability of that observation occurring given the action and the preceding

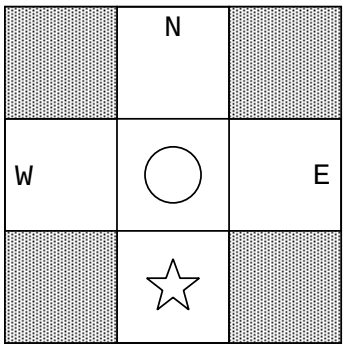


Fig. 1. Problem where an MDP-based model fails. The agent is in the center square, but in an unknown orientation. The goal is at the bottom.

state distribution. The end result of an action plus an observation is another state distribution. A POMDP policy specifies which action to take in each of the resulting state distributions. The optimal POMDP policy will choose the action with the highest value.

If the actions and observations are expanded in a search-tree form, they form an AND-OR tree.The nodes of the tree are state distributions. The actions form the OR branches, since the optimal action is a choice among the set of actions. The observations form the AND branches, since the utility of an action is a sum of the utility of the state distribution implied by each of the possible observations multiplied by the probability of that observation (see Equation 3).

If this tree were fully expanded to a depth d, the d-horizon POMDP value could be computed from this. The value of an OR node is the value of its child with maximum utility, weighted by the discount factor $\beta < 1$. The value of an AND node is the sum of the values of its children, each weighted by its probability. The subtree that corresponds to choosing the maximum utility child at each step is in fact the optimal policy for this horizon.

Since we want the optimal infinite-horizon policy, this technique is infeasible as stated. But we will work with a tree with estimated values at the fringe, which will give an estimated value for each branch of the tree. As the tree is expanded, the estimates will become more accurate (guaranteed by the discount factor β). The goal is to perform a search to find the policy with maximum utility.

If we invert the function so that the goal is to minimize disutility, we now have a classic search problem: given an AND-OR tree (the actions and observations) and an estimation function at the fringe, the goal is to find the path that minimizes the true function. Thus the AO* algorithm [7] can be used to expand the tree. Moreover, if the evaluation function is underestimating and monotonic, the algorithm is guaranteed not to return a non-optimal path as the goal. The AO* algorithm still involves a non-deterministic choice of which AND branch along the current best path to further expand (since that does not fall

out directly from the evaluations, unlike in A*). We will discuss a strategy for choosing the fringe node to expand in the next section.

5 Bounded, Incremental Search for POMDPs

The AO* algorithm by itself is incremental, in that after every fringe node expansion the best path represents the best current policy or plan. However, the AO* algorithm is limited by the memory it uses. In earlier tests, we ran into memory limitations after a small number of iterations, as the tree size grew quickly and filled memory.

To remedy this problem, BI-POMDP uses an iterative AO* algorithm [11] that uses memory linear in the depth of the search. This allows searches that are for practical purposes unlimited, as long as the fringe-node expansion strategy avoids extremely deep searches along a single branch.

The AO* algorithm does not specify the order for expanding children of AND nodes. BI-POMDP uses a strategy that chooses the node that presents the greatest potential to change the estimated value of the overall path. Since this node is only known with certainty when the true values are known, we use instead a heuristic estimate, which is the weighted difference between an underestimate and an overestimate of the node value. This represents the maximum possible change to the overall path value based on expanding this node. To use this heuristic estimate, we need two heuristic functions, one that underestimates, and one that overestimates.

To get an underestimating, monotonic heuristic function, we solve the (much easier) underlying MDP model of the problem domain. The solution describes the optimal policy and state values for the case of full observation. Tractable solution methods exist for MDPs, and this computation can take place off-line. The MDP solution is then used to produce a value function that is the sum of the MDP value of each state, weighted by the probability of being in that state:

$$V_F(\pi) = \sum_{i \in \mathcal{N}} \pi_i v(i)$$

(this is the approach suggested in [4] and [9]). This is an optimistic function, guaranteed to overestimate the utility [10]. Since node values are disutilities, we are minimizing disutility, and this optimistic function can be seen as an underestimating heuristic estimate of the disutility. In addition, it is straightforward to show that the function is monotonic.

To get the corresponding overestimating heuristic function, we compute the *worst-case* MDP. In particular, instead of choosing the action at each state that maximizes the utility, we choose the action that minimizes the utility:

$$v(i) = \min_{a \in \mathcal{A}} \left\{ \varrho_i(a) + \beta \sum_{k \in \mathcal{N}} v(k) p_{ik}(a) \right\}$$

This then produces the strategy that is guaranteed to produce the worst possible action for each state. These are then used in a value function like before, with

a weighted sum of the worst-case state values. This value function produces a pessimistic estimate that is guaranteed to be underestimate of the utility. So this function will produce an overestimate of the disutility, which is what we need. Note that this is a very crude bound, since in fact it produces a lower bound of the utility of the worst POMDP action, which is itself less than or equal to (usually significantly less than) the utility of the best POMDP action.

Note that by using a guaranteed optimistic and pessimistic estimate on the utility of a state distribution, we have not only an effective strategy for exploring useful parts of the search space, but we also have a bound on the possible utility of the current best action (see Figure 2). As the search proceeds, the upper and lower bounds narrow (since, after all, the search proceeds by expanding the branch with the largest gap). At the moment that the bounds meet, the policy is guaranteed to be optimal, since the optimal policy has a value between the upper and lower bounds. But earlier in the search it is possible to give a bound on the possible error of the value estimate: it is guaranteed to be in the range [lower bound, upper bound]. So this avoids the problem of following an action based on an overly optimistic estimate, only to find out that it was actually much worse.

6 Results

To test the BI-POMDP method, we have run it on a suite of problems from the POMDP literature. Here we present results from two of those problems, a one-trial version of the Tiger problem [4], and a robot navigation problem [10].

The Tiger problem is the choice between two doors, with a tiger behind one and money behind the other. A third action is to listen, gaining information but paying for the privilege. The one-trial version is simply that this choice is performed once, and then the process terminates (in a sink state). The listening may go on for an indefinite time, since conflicting information may be gathered from successive listens. The best case is that the money is chosen immediately. The worst case is that the tiger is chosen immediately. The optimal plan is to listen until enough certainty is accumulated, then choose. This is solvable by hand, although exact algorithms have difficulty with the one-trial version.

The robot navigation problem is a small corridor segment with alcoves. There is one alcove that is the goal, and the rest are highly penalized. The robot starts with ambiguity as to whether it is near the goal or another alcove. The best case is to move directly to the goal. The worst case is to move directly to another alcove. The optimal plan is to sense the world enough to determine whether it is near a goal or alcove, then move. This is small and reasonably obvious to the human eye, but unsolvable by current exact techniques.

For the Tiger problem, Figure 2(a) shows the bounds of the value estimate

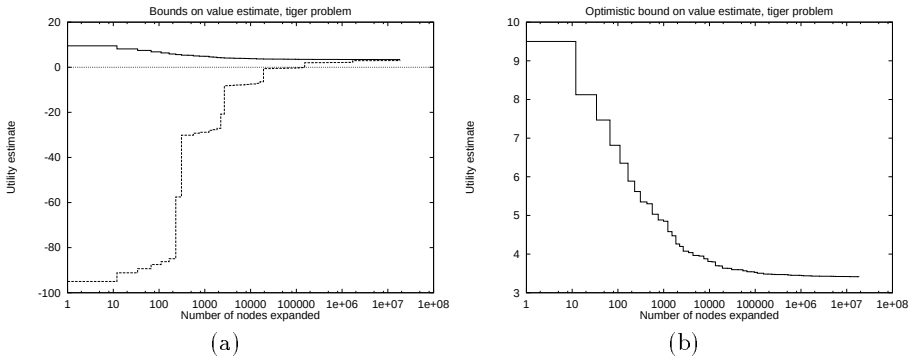


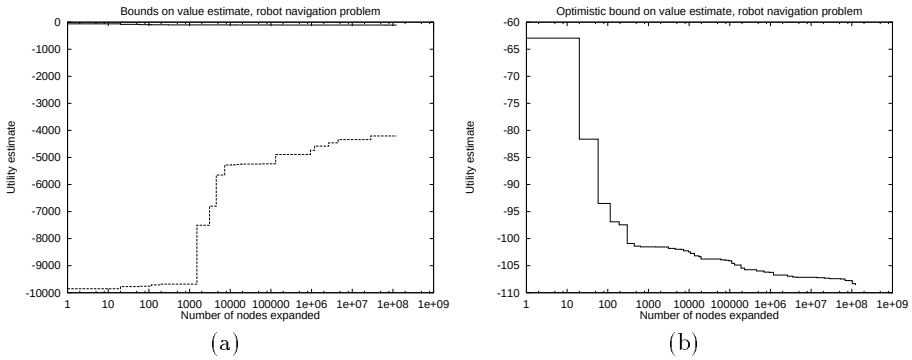
Fig. 2. Tiger problem. (a) Upper and lower bounds on the value of the best action, as a function of the number of nodes expanded[2]. (b) Optimistic bound on the value of the best action, as a function of the number of nodes expanded.

of the initial state distribution over time. As the search deepens, the bounds approach each other, reaching nearly provable optimality as they converge. To better show the evolution of the optimistic value estimate, which is fact the more accurate, Figure 2(b) isolates this. This in fact converges quickly towards the true value, but without an accurate pessimistic bound to help direct the search, it takes much longer for the bounds to converge.

For the robot navigation problem, Figure 3(a) shows the evolution of the bounds on the value estimate of the initial state distribution; Figure 3(b) shows the optimistic estimate. Note that in this case, the two bounds remain rather far apart, although the optimistic estimate is actually not far from the (hand-calculated) true value of about -116. Here the inaccurate overestimate proves to be a greater problem. In fact, the pessimistic bound is very bad in this case, since the worst possible action is to enter a highly-penalized alcove, of which there are plenty. A better pessimistic bound would help both the perceived value and the performance of the algorithm. Other problems in the suite of examples produce results varying between the two cases shown, with differences in rates of convergence.

Note that this is just one measure of plan quality, albeit an important one. Another important criterion, and perhaps the critical one, is the quality of the partial plan when it is executed. To test this, we constructed a version of the program that could be cut off after a given amount of time, after which the best known action would be executed, and the effects simulated stochastically. To test the quality of actions, this plan/execute cycle was repeated until the

[2] The plots are on a log scale in order to show the curve of convergence. This in fact makes the convergence look worse than it actually is—the bounds converge quickly at first and then asymptotically afterwards. When plotted on a linear scale, the graphs appear to show nearly immediate convergence followed by a straight line.



(a) (b)

Fig. 3. Robot navigation problem. (a) Upper and lower bounds on the value of the best action, as a function of the number of nodes expanded. (b) Optimistic bound on the value of the best action, as a function of the number of nodes expanded.

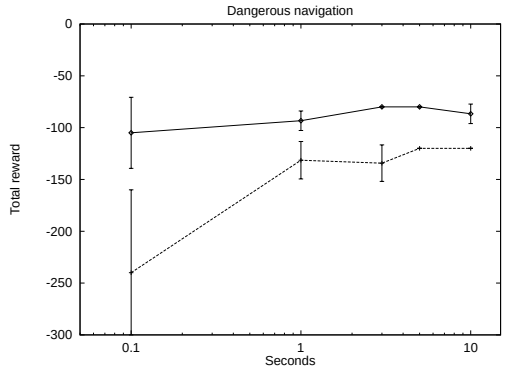


Fig. 4. Accumulated rewards from robot problem, as a function of the time cutoff. The two lines represent the cases where the robot starts near to or far from the goal.

process entered a stable state (sink or goal), and the total reward computed. We tested for a variety of time cutoffs, running 10 trials on each time cutoff.

The results for the robot navigation problem are shown in Figure 4. Note that when the robot is close to the goal, it does not improve much over time; this is because in fact it finds the optimal actions quickly (although more variation shows up with small time cutoffs). When the robot is far from the goal, the plan quality improves with the time allotted. In the Tiger problem, the plan quality remains flat because it is in fact too easy a problem and is solved nearly perfectly with minimal time expended. In both cases, the quality of the plan is somewhat hidden by the noise that exists from the stochastic simulation; for example, the reward may vary based on whether a robot motion succeeded or not.

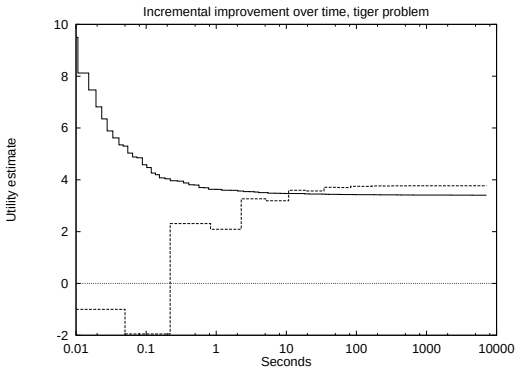


Fig. 5. Comparison of the convergence of BI-POMDP and Witness. Tiger problem. The curves cross due to small numerical inaccuracies accumulated by Witness and the underlying MDP estimate used by BI-POMDP.

7 Comparison to other POMDP approaches

Exact POMDP approaches are generally implemented as iterative algorithms that converge on the exact solution [4,6]. The algorithm generally regarded as the fastest is the Witness algorithm [2,4][3]. We compared the convergence of Witness and BI-POMDP on the two problems discussed here. Further comparisons are necessary (and planned) for other problems and also for the functional behavior of the approaches with respect to plan execution (what reward is actually received during execution of the incomplete plans).

For the Tiger problem, both the Witness and BI-POMDP approaches converge relatively quickly on the optimal value, as seen in Figure 5. In fact, the Witness algorithm did not converge to an exact solution after 2 hours of CPU time (Sun UltraSparc running Solaris).

For the robot navigation problem, the Witness algorithm terminated during the second iteration with a matrix inversion error. Monahan's algorithm warned of numerical instability during the second iteration, and then didn't finish the third iteration after more than two hours of CPU time[4]. Not surprisingly, the initial results of Witness and Monahan's algorithm are very inaccurate (-20 for Witness and -40 for Monahan, compared to the true value of approximately -116). In contrast, the BI-POMDP value converges to near the true value (starting at -63, -100 after 0.2 seconds), after which it converges more slowly. There remains some instability in the best action in the BI-POMDP case, since the bad

[3] A new algorithm by Zhang, Littman, and Cassandra has recently been released, but the program is still dependent on linear programming routines that are not public domain. A comparison to this new algorithm is forthcoming.

[4] This was using an implementation of Witness and Monahan's algorithms that relies on public-domain LP routines, which are more prone to numerical instability. The version of Witness described in published papers uses a superior, commercial LP product.

pessimistic bound forces the exploration of suboptimal search branches. However, the actions correspond to relatively innocuous moves, such as turning in place, instead of the optimal action of gathering more information to disambiguate the position of the robot.

8 Discussion

We have presented an incremental, search-based approximation for partially observable Markov models. The search is based on an incremental AND-OR search, using optimistic and pessimistic heuristic functions based on the underlying Markov model. In addition, the search provides a bound on the possible error of the approximation. In domains where a guarantee of safety is necessary, this bound is essential.

There remain limitations with this work. The current way of computing the pessimistic bound is very crude, and in fact the search strategy is hampered by the inaccuracy of this bound. A topic of current research is finding better pessimistic bounds, for instance using suboptimal policy graphs [3]. This will allow faster convergence and more useful information about the possible actions.

Partly because of the problem with pessimistic bounds, the search takes a long time to converge on a narrow bound. The policy itself may remain relatively stable, but the estimated value retains a wide bound.

The evaluation of the plan quality remains a problem. The ideal would be to compare the plan execution, as measured by accumulated reward, to the optimal plan execution. This however requires an optimal policy, and for those problems which would be most interesting to evaluate, an optimal policy remains out of reach.

Finally, other metrics for measuring the distance to optimal, such as policy similarity or functional behavior of partial policies, need to be investigated. In cases where the policy fluctuates, it is often due to nearly equivalent action choices, outweighed by the inaccuracies of the estimation functions. If the goal were to find a nearly optimal policy, eliminating those actions that are clearly worse, than perhaps a solution could be found more quickly with another metric.

9 Acknowledgements

Thanks to Michael Littman for supplying a suite of example problems and Tony Cassandra for the code for the exact POMDP methods. The work was started with support from a Veterans Affairs Medical Informatics fellowship, hosted jointly by the Philadelphia VAMC and the University of Pennsylvania; it was continued with a Chateaubriand Postdoctoral fellowship from the French government, hosted by the CRIN-CNRS & INRIA-Lorraine laboratory in Nancy, France.

References

1. R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.
2. A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of AAAI-94*, 1994.
3. M. L. Littman, 1996. Personal communication.
4. M. L. Littman, A. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In A. Prieditis and S. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 362–370, San Francisco, CA, 1995. Morgan Kaufmann.
5. W. S. Lovejoy. A survey of algorithmic methods for partially observed markov decision processes. *Annals of Operations Research*, 28:47–65, 1991.
6. G. E. Monahan. A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.
7. N. J. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing Company, 1980.
8. R. Parr and S. Russell. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of IJCAI-95*, 1995.
9. R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of IJCAI-95*, 1995.
10. R. Washington. Incremental Markov-model planning. In *Proceedings of TAI-96, Eighth IEEE International Conference on Tools With Artificial Intelligence*, 1996.
11. R. Washington. IDAO*: Incremental admissible AND-OR search. Draft paper, 1997.