A Heuristic Variable Grid Solution Method for POMDPs

Ronen I. Brafman
Department of Computer Science
University of British Columbia
Vancouver, B.C. V6T 1Z4 Canada
brafman@cs.ubc.ca
http://www.cs.ubc.ca/spider/brafman

Abstract

Partially observable Markov decision processes (POMDPs) are an appealing tool for modeling planning problems under uncertainty. They incorporate stochastic action and sensor descriptions and easily capture goal oriented and process oriented tasks. Unfortunately, POMDPs are very difficult to solve. Exact methods cannot handle problems with much more than 10 states, so approximate methods must be used. In this paper, we describe a simple variable-grid solution method which yields good results on relatively large problems with modest computational effort.

Introduction

Markov decision processes (MDPs) (Bellman 1962) provide a mathematically elegant model of planning problems where actions have stochastic effects and tasks can be process oriented or have a more complex, graded notion of goal state. Partially observable MDPs (POMDPs) enhance this model, allowing for noisy and imperfect sensing, as well. Unfortunately, solving POMDPs, i.e., obtaining the optimal prescription for action choice at each information state, is a computationally daunting task, feasible only in extremely small domains. For example, the Witness algorithm (Kaelbling, Littman, & Cassandra 1996), one of the best exact solution algorithms, can handle problems with about 10 states, and is not a realistic alternative in larger domains (Littman, Cassandra, & Kaelbling 1995a).

Consequently, attempts have been made to come up with methods for obtaining approximately optimal policies (e.g., (Lovejoy 1991a; Littman, Cassandra, & Kaelbling 1995a; Parr & Russell 1995)). It is hoped that a combination of good approximation algorithms, structured problems, and clever encodings will lead to acceptable solutions for realistic POMDPs. In this paper, we describe a variable grid algorithm for obtaining approximate solutions to POMDPs in the infinite horizon case, which shows some promise.

It is well known (Sondik 1978) that an optimal policy for a POMDP with n states can be obtained by solving the *belief-space* MDP whose state space consists of all probability distributions over the state space of the POMDP, i.e., an n-dimensional simplex. Grid approximations attempt to solve the belief-space MDP directly by placing a discrete, finite grid on this n-dimensional space and restricting their calculations to the points of this grid. The computational effort required for approximating the value function on a k point grid is roughly equivalent to that of solving a k state MDP. Fixed grid approximations (e.g., (Lovejoy 1991a)) construct a grid based on the size of the state space alone. Hence, they are (almost) problem independent. Grid construction is simple and fast, and the regularity of the grid facilitates simple value interpolation algorithms. Variable grid methods use a problem dependent grid that may change during the solution process. They allow us to concentrate our effort where it is most needed and to obtain good approximations with a much smaller grid. However, interpolation is harder to perform, and some variable grid construction methods (e.g., (Cheng 1988)) are quite complex, requiring considerable computation time and space.

In this paper, we describe a simple variable grid method that performs well on a number of test problems from (Littman, Cassandra, & Kaelbling 1995a). Our experience indicates that this method requires modest computational effort; thus, we believe that it is a realistic alternative for solving large POMDPs. Our approach is motivated by the following hypothesis: (1) Variable grids can lead to considerable efficiency, allowing us to gain more information where it is needed most while minimizing grid size. (2) Rough solutions provide heuristic information that can be used to construct better grids, suggesting an iterative approach. In addition, we observe that computationally feasible variable grid methods call for a simple grid construction method.

Roughly, our algorithm operates as follows:
(Step 1) Use the underlying MDP to generate an initial grid and an initial estimate of the value function.
(Step 2) Use current estimate to generate new grid points.
(Step 3) Use the new grid to estimate the value function.
(Step 4) If desirable/feasible, go back to step 2.
The final estimate of the value function is used to generate a policy. In our experiments with an 89 state POMDP described in (Littman, Cassandra, & Kaelbling 1995a), we were able to obtain very good results using a 337 point grid. In contrast, the coarsest grid required by the fixed grid method of (Lovejoy 1991a) contains 3916 points.

This paper is organized as follows: Section 2 discusses related grid-based methods; Section 3 describes our algo-

rithm; Section 4 presents our experimental results; And Section 5 concludes the paper. We assume familiarity with MDPs and POMDPs. For good introductions, consult (Kaelbling, Littman, & Cassandra 1996; Lovejoy 1991b).

Grid-Based Methods

Given an n state POMDP, its corresponding belief-space MDP has an uncountably large state space, consisting of an n dimensional simplex. Hence, it cannot be solved by standard MDP solution algorithms. Grid-based solution methods attempt to approximate the value function over the entire state space by estimating it on a finite number of belief states, those contained on the chosen grid. There are three main dimensions along which grid methods differ: (1) How the grid points are generated; (2) How the value function (or its gradient) is estimated on the grid; (3) How value estimates on grid points are generalized to the whole space. In what follows, we discuss some approximation methods that are closely related to our approach, noting how each addresses the three questions above. A more complete description of approximation methods appears in the full paper, or see (Lovejoy 1991b; White 1991).

The Q_{MDP} method of (Littman, Cassandra, & Kaelbling 1995a) can be viewed as a grid based method in which the grid consists of the states of the POMDP. The underlying MDP is solved, and its value and Q functions are used as estimates for the grid points (which correspond to the states of the underlying MDP). For a non-grid belief state b and an action a, one estimates $Q(b, a)$ using the inner product of the vectors b and q_a. Here, q_a is the vector of Q values for the states in the underlying MDP (i.e., the grid points) and the action a. This method performs relatively well in a number of navigation domains introduced in that paper. However, it causes the agent to act as if it will end up in a state of perfect information, i.e., a belief state corresponding to one of the underlying MDP's states. Consequently, the agent never executes an action whose sole effect is to obtain information. Our method will relax this assumption by allowing the agent to act as if it can end up in some other belief states — those that are part of our grid.

It is known that the optimal value function V_* of an MDP is the fixed point of the dynamic programming update operator H, defined as:

$$(Hv)(s) = \max_{a \in \mathcal{A}} R(s, a) + \sum_{s' \in \mathcal{S}} T(s, a, s') \cdot v(s').$$

Here, v is a value function, $\mathcal{A}$ is the set of actions, $\mathcal{S}$ is the set of states, R is the reward function, and T is the transition function. Through repeated applications of H to some random initial value function, we can approach V_*. Cheng's *iterative discretization procedure* (IDP) (Cheng 1988) solves POMDPs using an approximate version of his linear support algorithm, which is an exact algorithm for performing dynamic programming updates. The linear support algorithm creates a set of polygonal regions such that there exists a single action whose Q value is maximal on all belief states within this region. IDP works as follows: Given a value function v_n, we apply the operator H', obtained when we terminate the linear support algorithm prior to completion, obtaining a new estimate v_{n+1}. This calculation is not grid based, and repeating it a number of steps, although an option, is not likely to be practical. Instead, Cheng applies a discrete approximation step, in which the value is updated only for some set of points, rather than the whole space. As grid points, he suggests using the set of vertices of the regions created by the previous step in which H' was applied; hence, this is a variable grid. A set of gradients to the value function, G, is generated for the grid points, and the value function on a belief state b is defined as $\max_{g \in G} b \cdot g$. Cheng's algorithm alternates one H' step with one discrete approximation step. An error estimate can be obtained, although it requires some computational effort.

Lovejoy (1991a) develops a fixed-grid method which calculates upper and lower bounds on the optimal value function. Using these bounds, we can get an approximately optimal policy and an error estimate. The method is geared towards finite horizon problems, but an extension to the infinite horizon case is considered. Lovejoy does not insist on any particular grid for generating the lower bound. However, given the grid and using a standard method, he generates gradients for the value function at the grid points. As hinted above, a set of gradients to the value function at any set of points can be used to approximate the value function everywhere. Moreover, if the estimate on grid points is a lower bound, the value function obtained is a lower bound for the entire space. For the upper bound, Lovejoy uses the Freudenthal triangulation (Eaves 1984) to construct a particular grid which partitions the belief space, the simplex of order $|\mathcal{S}|$, into sub-simplices. Using dynamic programming updates, the value function is estimated on the grid points. Lovejoy's choice of grid allows for an elegant and efficient interpolation method which estimates the value of arbitrary belief states based on the value of the grid points in the smallest sub-simplex containing this state. Because the value function is continuous and convex, one can obtain an upper bound on the optimal value function using this method (assuming the estimates at the grid points are upper bounds). Lovejoy reports good results on POMDPs with approximately 10 states.

Our method is iterative and uses discrete approximation steps like Cheng's IDP, although much simpler ones. It updates the value function on grid points using dynamic programming updates, much like Lovejoy's method for obtaining upper bounds. In fact, our method yields an upper bound on the optimal value function as well. However, because we use a variable grid, a simpler, and probably weaker, interpolation method is used to generalize these estimates to the whole grid.

Generating approximating MDPs

Our method works by generating a sequence of grid-based, approximate value function for the belief space MDP. Each function is generated by assigning values to a number of belief states – the grid points – which are then interpolated to obtain an assignment for all other points. Naturally, the interpolated values are not stored explicitly, but calculated

when needed. The first function is obtained by choosing the states of the underlying MDP as the grid points. Each following function is generated by adding new points to the grid, recalculating their values, and interpolating to non-grid states. The decision as to whether or not to add a particular belief state to the grid is made online based on: (1) the importance of obtaining a better estimate of its value, assessed using the current value function and its induced policy, and (2) the likelihood of reaching this state. Hence, states are added in an informed, gradual manner. It is hoped that using this approach, we shall obtain progressively better estimates without adding too many states.

The basic process we follow is similar to Cheng's IDP: (1) Generate a grid. (2) Use the grid to (implicitly) generate a value function. (3) Depending on resource constraints (i.e., time and memory) and needs, use the current value function to generate a new, finer grid, repeating the process. Finally, the last (implicitly defined) value function is used to generate a policy, as needed.

The initial grid generated contains all perfect information belief states, which correspond to the states of the underlying MDP. Their value is estimated as the value of the corresponding states in the underlying MDP. This estimate is expanded to the whole space using an interpolation method which, at this stage, generates the same value Q_{MDP} generates. Hence, our initial estimate is identical to that obtained by Q_{MDP}.

In order to complete our method's description, we must explain: (1) the method used to add new grid points; (2) the interpolation method. We start with the first question.

Given the current value function, we wish to generate a new, finer grid. We add new belief states to the current grid by examining combinations of existing belief states (usually pairs) and considering whether more information within their convex hull would be useful. There are two considerations: how different the policy is likely to be within the region, as opposed to the boundary of the region; and how likely are we to reach states within that region. To illustrate the first point, consider two arbitrary points b_1, b_2 in belief space and suppose, for the moment, that we are considering a one step horizon. If in both states the optimal action is a then a is the optimal action on all points on the segment connecting b_1 and b_2. If a_1 is the best action on b_1, and a_2 is the best action on b_2, but both $|Q(a_1, b_1) - Q(a_1, b_2)|$ and $|Q(a_2, b_1) - Q(a_2, b_2)|$ are very small, both a_1 and a_2 are close to optimal anywhere between b_1 and b_2. In these two cases, there is little motivation to explore this region farther. If, on the other hand, the above differences are not negligible, more attention should be paid to this region. This argument is not generally valid: when we consider an horizon of 2, we must compare two-stage decision trees instead of single actions; and when we consider infinite horizon problems, we must compare infinite-stage decision trees. However, it provides a useful heuristics.

The second consideration we employ is that of reachability. Even when the above Q value differences for the b_1 and b_2 are not negligible, it may be the case that it is unlikely that we reach a belief state which is in between b_1 and b_2. A full fledged reachability analysis is likely to be computationally costly and may generate a vast number of states (although see Section 5 for some results along this line). Instead, we use a simpler, local method: we can conclude that it is unlikely to reach a state in between b_1 and b_2 if the messages we are likely to receive on each of these states are different.

Our current implementation uses the above ideas as follows. For every pair b_1 and b_2 of grid states, the Q value differences discussed above are calculated based on the current value function. If these differences cross some threshold value, we consider adding the mixture state $b_{12} = 0.5 \cdot b_1 + 0.5 \cdot b_2$. We assess the likelihood of reaching b_{12} by examining the probability of obtaining the same message on both b_1 and b_2.[1] If this likelihood is greater then some threshold, we add b_{12} to the set of grid points.[2] Naturally, instead of examining pairs of states, we could examine triplets, and so on, using identical considerations. Indeed, in some experiments reported here we have done this.

Having generated a new grid, we must generate a new estimate for the value function. This itself is an iterative process similar to the standard value iteration algorithm (Bellman 1962). The value function is updated on the grid points using standard dynamic programming update, as in any MDP solution method. This update process is repeated until the value function for the grid points stabilizes (i.e., the difference between the value functions obtained on two consecutive iterations is smaller than some fixed parameter).

Typically, the update operation will require estimating the value function on non-grid points. In addition, once the value for grid points is set, we must be able to generate a value function on the whole belief space if we wish to be able to assign actions in arbitrary belief states. We considered two methods for assigning values to non-grid points. The first method used the value of the nearest neighbor (based on k^{th}-norm distance in belief space, for $k = 1, 2$). This method faired poorly, and we will not present more detailed results of its performance (but see Section 4 for a limited application of the nearest neighbor method). The second method is an interpolation method in which a belief state b is represented as a linear combination of grid states with positive coefficients. When the value function on grid points is an upper bound on the optimal value function (as is the case in our approach), this method generates an upper bound on the optimal value function on all states. Ideally, we would like to use grid points closest to b to perform this interpolation, as this would yield the best approximation. (This simple observation follows from the fact that our estimates are upper bounds and that the optimal value function is convex.) Because our grid is not fixed, we cannot use Lovejoy's interpolation method which does just that. However, we have come up with a reasonable alternative in which linear combinations containing grid states in which the agent has less information are preferred over more informative states. Such states will be closer, typically, to the

[1] Actually, we must also consider the action performed, and the discussion above extends naturally to that case.

[2] In fact, in the experiments reported here, we added b_{12} only if the most likely message in b_1 and b_2 is the same.

target state b.

To obtain this preference, we do the following: the grid points are ordered according to the amount of information they encode. States with more complete information come first, followed by states with less information (e.g., as defined by their entropy). Given a belief state b, the list of grid points is searched backwards, until a belief state b_{g_1} on the grid is found that assigns positive probability only to states to which b assigns positive probability. The maximum coefficient c_1 for which $(b - c_1 \cdot b_{g_1} \geq 0)$ is calculated, and the process continues with $b - c_1 \cdot b_{g_1}$ instead of b. This method, while not perfect, is computationally simple and ensures that states close to b are used in the interpolation. Unlike the nearest neighbor approach, the value obtained takes into account all positive probability states. Notice that if we prefer states with more information, i.e., if we search the grid points forward, we end up with the approximation methods used in Q_{MDP}.

Given the final value function, the policy on a belief state b is generated as follows: the Q value of each action on b is estimated using the interpolation method based on the Q values of the grid states. The action whose Q value is best is chosen. Some variants of this methods are discussed as well.

Experimental Results

Assessing the suitability of techniques for generating policies for POMDPs is not a simple task. There are no analytical or computational methods for generating optimal solutions for POMDPs with more than about 10 states; hence, we cannot compare generated policies to the optimal policy. One option is to supply an error estimate, and this is possible by emulating Lovejoy's method and generating a lower bound. However, we want to assess the quality of our current heuristic method. We suspect that as the problems we examine become more complex, useful solution methods will be of this nature and error bounds will either be too lax or will require considerable computational effort (e.g., as in Cheng's IDP or when seeking uniform bounds in Lovejoy's method). We shall resort to simulations as a means of assessing the quality of the proposed solution method. Even this approach runs into some difficulties because there is little experience with large state spaces and few results to which we can compare ours. The only two attempts to experiment with large POMDPs we are aware of are those of (Littman, Cassandra, & Kaelbling 1995a) and (Cassandra, Kaelbling, & Kurien 1996), and we examine the performance of our method on two of their domains.

Figure 1 describes the largest test problem examined in (Littman, Cassandra, & Kaelbling 1995a) (in fact, this is the largest test problem we have seen). In this problem, a robot must reach some goal position starting from a random initial state. At each position, the robot may have one of four possible orientations and can take one of five actions: stay-in-place, move-forward, move-right, move-left, turn-around. The effects of these actions are stochastic. Following each action, the robot receives information about the configuration of walls around it. However, the sensors are noisy and

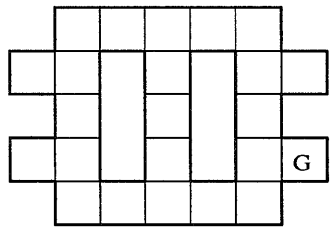

Figure 1: An 89 state maze

may erroneously indicate a wall when one does not exist and vice versa. A more precise description of this set-up, including the transition probabilities and message probabilities appears in (Littman, Cassandra, & Kaelbling 1995b). The initial belief state of the robot is uniform over all possible robot configurations (i.e., position-orientation pairs) outside the goal. Upon reaching the goal the robot receives a reward of 1. The discount factor is 0.95.

In (Littman, Cassandra, & Kaelbling 1995a), the best results for this domain were obtained by the Q_{MDP} method with the stay-in-place action suppressed. (Equally good results were obtained using Q-learning based methods which we do not consider.) They conducted 251 experiments, in each of which the robot was given a maximum of 251 steps to reach the goal state. They recorded the percentage of experiments in which the goal was reached and the median length of each experiment. Interestingly, in this domain, with little training, human performance is much better. In (Cassandra, Kaelbling, & Kurien 1996), a number of additional methods for solving navigation POMDPs were proposed. In particular, the authors recommend a method in which the agent performs the action that is assigned by the underlying MDP to the most likely state (MLS). We note that this method is an instance of the nearest neighbor method, which we discuss later.

First, we present the results we obtained using our method on three grid sizes when we used the interpolation method to estimate the Q values of actions on each belief state. The first grid is generated by the underlying MDP. In this case, our method is equivalent to the Q_{MDP} method. The second grid contains 159 states which were obtained by adding belief states as described earlier. We added pairs of states which had the same most likely message, were assigned different actions by the underlying MDP, and their Q values differed on the pair of preferred actions by more than 0.1 (see the earlier discussion). Another grid contained 337 states which were obtained as above, but while removing this latter restriction on the Q values and allowing combinations of three states, as well.

Our results are described in Table 1. *Standard* refers to the interpolation method discussed earlier; *standard w/o STAY* refers to the standard method when we do not consider the stay-in-place action. With both of these methods we can see significant improvement as the state space increases. In particular, with modest increases, we obtain results that substantially outperform the Q_{MDP} method and are better than the MLS method (discussed later).

Size	Estimation Method	Success	Median
–	human	100%	29
89	Q_{MDP} (= standard)	27%	> 251
159	standard	65%	36
337	standard	98%	24
89	Q_{MDP} w/o STAY	58%	40
159	standard w/o STAY	76%	31
337	standard w/o STAY	100%	24

Table 1: Results for 89 state maze using the interpolation method

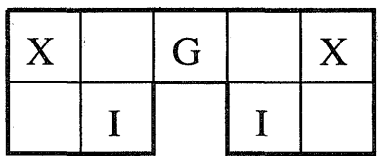

Figure 2: A 33 state maze with negative rewards

The use of the standard method w/o stay-in-place may seem unmotivated. In fact, it can be viewed as a crude implementation of a general heuristic that overcomes an inherent problem of our interpolation method: loops. The interpolation method uses the Q values of different grid points to estimate the Q values of a particular belief state. In MDP solution algorithms, only the relative size of Q values is important. In our approximation algorithms, the actual value is used in the interpolation. However, $Q(b, a)$ only approximates the values we need for a good interpolation – the value of the "mutated" policy, $p[a \rightarrow b]$, obtained from the current policy, p, when a is assigned to b. Notice that if a is a no-op on belief state b, as stay-in-place is for certain belief states, the value of $p[a \rightarrow b]$ in b will be 0 or close to 0.[3] Yet, $Q(b, a)$ will be close to the value of b in the original policy p. Thus, the Q values are somewhat misleading.

In our particular case, different motion actions will seem good on some states and bad on others. But stay-in-place always seems harmless, judging from its Q value. Hence, when there is sufficient uncertainty, its value is the highest. However, in many such states, it is really a no-op and loops back to the same belief state. If we implement loop detection, we can overcome this problem.[4]

Another domain examined in (Littman, Cassandra, & Kaelbling 1995a) is shown in Figure 2. While this is a much smaller problem, it was set up to bring out the need for actively seeking information. The goal state is marked G; the two possible initial states are marked I; the agent may start in any initial state facing upwards; the initial belief state assigns equal probability to each of the two possible initial configurations; the two squares marked X represent 8 states (two positions $\times$ four headings) in which a negative reward of -1 is obtained. When the goal is reached, a reward of 1 is given and the agent randomly moves to one of the initial states.

In both initial configurations, moving forward is the best action under perfect information. However, moving forward is not a good idea in the initial belief state. By moving forward, the agent reaches the two upper positions adjacent to the goal. It cannot differentiate between these positions, since they generate the same sensor readings. Hence, it does not know what to do. Depending on where it is, a move in each direction could either take it to the goal or to a state in which negative reward is generated. The right strategy is to sense first, by performing the stay-in-place action. This will tell the agent where it is. Then, it will know what to do after it moves forward. Our results are described in Table 2. The underlying MDP has 33 states. The 82 point grid contains states corresponding to belief states in which various pairs of (underlying) states are equally likely. The 174 state MDP contains additional combinations containing more than two possible states.

Because reaching the goal quickly is not the only objective here, we present the average discounted reward (ADR) obtained for each method over 151 experiments, each lasting 500 steps. (STD is the standard deviation.) Again, we can see improved performance with larger approximating MDPs. Yet, the interpolation method does not offer good results. However, when combined with one lookahead step, it gives excellent results, which seem to be close to optimal. The lookahead step is employed as follows: given the current belief state, for each action we determine the set of possible following belief states and their probability. The values of these belief states are estimated as described previously, and an expected value is calculated for the action. The action leading to a state with maximal value is chosen. This is to be contrasted with our regular method which estimates the Q values of the current belief states based on the Q values of grid points.

The success of the interpolation method combined with lookahead is not surprising. With one lookahead step, the need for making an initial observation is revealed and we are able to act accordingly. This need is not foreseen by the standard interpolation method because of our choice of states. Recall that if two grid points are assigned the same action by the current estimate, we do not add their mixture to the next grid. In our case, both initial states are assigned the move-forward action by the optimal policy of the underlying MDP, and hence, we do not add them to any of the larger grids. However, it is important to notice that the one-lookahead step is not useful unless we have a richer state space. As can be seen, the Q_{MDP} method combined with one-lookahead performs poorly.

Notice that the move from the 82 point grid to the 174 point grid does not enhance solution quality. The reason for this phenomena is that in this domain, in most reachable belief states only two underlying states are highly likely. (This is a result of the particular initial state used and the nature of actions and observations in this domain). The 82 point grid

[3] Since a is a no-op on b, we will be stuck in b, obtaining no reward. The actual value depends on whether a has some chance of escaping b, in which case, we will not be stuck in b forever.

[4] Loop avoidance is not always a good idea, e.g., in process oriented domains and when a no-op causes the least harm.

Size	Estimation Method	ADR	STD
33	Q_{MDP} (=standard)	0.07	0.29
82	standard	0.8	1.35
174	standard	0.94	1.14
33	Q_{MDP} w/o STAY	0.04	0.4
82	standard w/o STAY	0.7	1.66
174	standard w/o STAY	0.7	1.28
33	standard + lookahead	0.04	0.40
82	standard + lookahead	2.35	1.21
174	standard + lookahead	2.27	1.01

Table 2: 33 state maze using interpolation method

contains all belief states with two probable POMDP states that our algorithm would consider; the 174 point grid contains additional mixtures, mostly of three underlying states, which, because of our interpolation method, have little influence on the generated estimates. When the nearest neighbor method is used (see Table 4), these states do come into play, and we see a modest improvement when moving into a larger grid.

An alternative approach for using the grid point's values to generate a policy is the nearest neighbor technique.[5] Given a belief state, we search for its nearest grid point, and we imitate its behavior. Various metrics can be used, as well as variants, such as k nearest-neighbor. Surprisingly, the nearest neighbor method works quite well in the larger domain when we use the underlying MDP, in which case it is equivalent to the MLS method mentioned earlier. It never fails to reach the goal, and the median number of steps is 45. Moreover, it requires little computation. However, it is clear that this method is limited in its scope to domains in which errors are recoverable and do not carry a large penalty. When these assumptions are false, as in the smaller domain which contains negative rewards, it performs poorly. There, its average reward is -0.76, compared to over 2 for the better methods. Moreover, even in the large POMDP, its average reward is less than half the optimal reward, and with a lower discount factor, its relative performance is even worse.

We experimented with the nearest neighbor technique using different metrics, and in general it was inferior to the interpolation method. Figures 3 and 4 contain some of the results we obtained when using 1-norm distance in belief space. A longer version of this paper contains additional results using this method and an explanation of the observed decline in solution quality between the 89 and 159 point grids).

Discussion

We presented a simple, heuristic variable-grid method for generating policies in stochastic domains with partial information. On existing domains, this methods performs very well with reasonable effort. We believe that our results constitute proof-of-concept and indicate that the solution approach presented here is worth studying. Even in its current form, our method can be used for investigating much larger domains. For example, obtaining a solution using the 337 point grid for the 89 state POMDP required roughly 800 second of CPU time on a Sun UltraSPARC 2 using our unoptimized algorithm.

We anticipate that additional refinements of the grid construction method will lead to methods in which grid construction is more informed (e.g., by using information about the initial state, or through better use of current information) and value interpolation is more efficient (e.g., by choosing an even better set of points as a basis for value function estimation). In addition, general heuristics, such as loop elimination, can be used to improve the policy generated. Such enhancements together with code optimization can lead to much more powerful and efficient methods.

Indeed, initial experiments with two other techniques for generating grids were quite positive. In particular, we examined the use of reachability analysis and lookahead. In reachability analysis the grid is constructed based on the set of belief states reachable from the initial belief state. If this set of states is small enough, it would be an ideal candidate for the set of grid points. This is essentially the method used in (Sondik & Mendelssohn 1979) for solving small POMDPs. Unfortunately, when the POMDP is large and there is sufficient uncertainty in actions' effects, the set of reachable states is very large. In that case, methods must be developed that allow us to integrate some partial form of reachability analysis into the grid generation process. In the 89 state grid discussed in this paper, we have been able to obtain very good results using an ad-hoc method that considerably limited the number of reachable states. (We only considered belief states in which the same probability was assigned to some set of underlying states while 0 probability was assigned to all other states.) With only 175 grid points we were able to obtain a success rate of 97% with median of 27 steps to the goal using the interpolation method (with the stay-in-place action). This is much better than the

Size	Estimation Method	Success	Median
89	MLS (= nearest neighbor)	100%	45
159	nearest neighbor	48%	>251
337	nearest neighbor	97%	40

Table 3: 89 state using nearest neighbor

Size	Estimation Method	ADR	STD
33	MLS (= nearest neighbor)	-0.73	1.92
82	nearest neighbor	0.45	1.43
174	nearest neighbor	0.61	1.38

Table 4: 33 state using nearest neighbor

[5] Nearest neighbor can be used as a general technique for extending the value function from grid points to the whole space. As remarked earlier, we have found that using this method during the initial stages of grid construction leads to poor performance.

Size	Estimation Method	Success	Median
159	standard	65%	36
184	standard	95%	29
197	standard	97%	30
159	standard w/o STAY	76%	31
184	standard w/o STAY	99%	29
197	standard w/o STAY	98%	28

Table 5: Using lookahead to choose new grid points

results described earlier, and it indicates that the 89 state maze, despite being larger than other test domains, is still not challenging enough. In larger domains, the set of reachable states, even of this form, will be too large to be used directly and more sophisticated techniques for reachability analysis will be needed. In addition, it should be noted that reachability analysis is quite costly! It can require as much or greater time and space resources than the solution phase. Hence, using a larger, more easily generated grid is sometimes a better alternative.

Another method we used, based on insights gained from the earlier experiments, uses the following technique to determine whether or not to add a belief state to the grid: We estimate the value and best action for the candidate state using the current value function and compare them to the estimate obtained for this state when it is added to the current grid. If these two estimates are significantly different, we add the state to the grid. Essentially, this method compares the current estimate of the candidate state with a lookahead estimate. This method works poorly when the current estimate is poor. However, once we have a reasonable value function, it seems to be making good choices. Table 5 shows the improvement in performance as new states are added based on this method when we start from the estimate obtained earlier on based on a 159 point grid.

One problem that is yet to be addressed in our approach is the handling of information. The domains on which we experimented are such that sensing is performed instantaneously and "for free" following each action. While this is a reasonable model for certain robots, it is not a very general one. The less information we obtain for free, the larger the grid we would generate. We hope that this problem, too, can be handled by an iterative process. First, a grid would be built based on the assumption that information is instantaneous and "for free." This assumption will be gradually weakened as we generate subsequent grids. However, this is still unexplored territory.

Finally, we must caution that good understanding of heuristic methods such as ours requires examining diverse large POMDPs. Clearly, the robot navigation environments we have used have special properties that may give us a skewed picture of an algorithm's performance. In particular, these domains exhibit much symmetry, they are goal oriented, actions are reversible, and there is little flexibility in the manner in which goals can be achieved. Indeed, MLS is not a good general method, yet it performs well on some such domains. We believe that the method we presented is applicable in diverse domains, but only further study can prove this. We hope that the growing interest in POMDPs will lead to the generation of new, diverse test problems that can be used to make better assessments of various solution techniques.

Acknowledgment: I wish to thank Tony Cassandra and Michael Littman for providing me with their test data, much advice, and references. I am also grateful to Craig Boutilier and David Poole for their useful comments and suggestions. My work is supported by IRIS project IC-7 and NSERC grants OGPOO44121 and A9281.

References

Bellman, R. E. 1962. *Dynamic Programming*. Princeton University Press.

Cassandra, A. R.; Kaelbling, L. P.; and Kurien, J. 1996. Acting under uncertainty: Discrete bayesian models for mobile-robot navigation. In *Proc. Int. Conf. on Intelligent Robots and Systems*.

Cheng, H. 1988. *Algorithms for Partially Observable Markov Decision Processes*. Ph.D. Dissertation, University of BCritish Columbia. School of Commerce.

Eaves, B. C. 1984. *A course in Triangulations for Solving Differential Equations with Deformations*. Berlin: Springer-Verlag.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1996. Planning and acting in partially observable stochastic domains. Submitted to Artificial Intelligence.

Littman, M. L.; Cassandra, A. R.; and Kaelbling, L. P. 1995a. Learning policies for partially observable environments: Scaling up. In *Proc. of the 12th International Conf. on Machine Learning*.

Littman, M. L.; Cassandra, A. R.; and Kaelbling, L. P. 1995b. Learning policies for partially observable environments: Scaling up. Technical Report CS-95-11, Dept of CS, Brown University.

Lovejoy, W. S. 1991a. Computationally feasible bounds for pomdps. *Operations Research* 39(1).

Lovejoy, W. S. 1991b. A survey of algorithmic techniques for pomdps. *Annals of OR* 28.

Parr, R., and Russell, S. 1995. Approximating optimal policies for partially observable stochastic domains. In *Proc. Fourteenth International Joint Conference on AI.*

Sondik, E. J., and Mendelssohn, R. 1979. Information seeking in markov decision processes. Technical report, Southwest Fisheries Center, National Marine Fisheries Service, NOAA, Honolulu, HI.

Sondik, E. J. 1978. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations Research* 26(2).

White, C. C. 1991. A survey of solution techniques for pomdps. *Annals of OR* 32.