POMDP Planning for Robust Robot Control

Joelle Pineau
School of Computer Science
McGill University
Montreal QC CANADA H3A 2A7
jpineau@cs.mcgill.ca

Geoff Gordon
Center for Automated Learning and Discovery
Carnegie Mellon University
Pittsburgh PA 15232
ggordon@cs.cmu.edu

Abstract

POMDPs provide a rich framework for planning and control in partially observable domains. Recent new algorithms have greatly improved the scalability of POMDPs, to the point where they can be used in robot applications. In this paper, we describe how approximate POMDP solving can be further improved by the use of a new theoretically-motivated algorithm for selecting salient information states. We present the algorithm, called PEMA, demonstrate competitive performance on a range of navigation tasks, and show how this approach is robust to mismatches between the robot's physical environment and the model used for planning.

1 Introduction

The Partially Observable Markov Decision Process (POMDP) has long been recognized as a rich framework for real-world planning and control problems, especially in robotics. However exact solutions are typically intractable for all but the smallest problems. The main obstacle is that POMDPs assume that world states are not directly observable, therefore plans are expressed over *information states*. The space of information states is the space of all beliefs a system might have about the world state. Information states are easy to calculate from sensor measurements, but planning over them is generally considered intractable, since the number of information states grows exponentially with planning horizon.

Recent point-based techniques for approximating POMDP solutions have proven effective for scaling-up planning in partially observable domains [5, 10, 11]. These reduce computation by optimizing a value function over a small subset of information states (or *beliefs*). Often, the quality of the solution depends on which beliefs were selected, but most techniques use ad-hoc methods for selecting beliefs.

In this paper, we describe a new version of the point-based value approximation which features a theoretically-motivated approach to belief point selection. The main insight is to select points which minimize a bound on the error of the value approximation. This allows us to solve large problems with fewer points than previous algorithms, which leads to faster planning times. Furthermore because a reachability analysis is used to select candidate points, we restrict the search to relevant dimensions of the belief, thereby alleviating the curse of dimensionality.

The new algorithm is key to the successful control of an indoor mobile service robot, designed to seek and assist the elderly in residential environments. The experiments we present show the robustness of the approach to a variety of challenging factors, including limited sensing, sensor noise, and inaccurate models.

2 Background

The Partially Observable Markov Decision Process (POMDP) provides a general framework for acting optimally in partially observable domains. It is well-suited to a great number of robotics problems where decision-making must be robust to sensor noise, stochastic controls, and poor models. This section first establishes the basic terminology and essential concepts pertaining to POMDPs.

2.1 Basic POMDP Terminology

We assume the standard formulation, whereby a POMDP is defined by the n-tuple: $\{S, A, Z, b_0, T, O, R\}$. The first three components, S A and Z denote finite, discrete sets, where S is the set of states, A is the set of actions, and Z is the set of observations. In general, it is assumed that the state at a given time t, s_t, is not observable, but can be partially disambiguated through the observation z_t. The next three quantities, b_0, T, and O define the probabilistic world model that underlies the POMDP: b_0 describes the probability that the domain is in each state at time $t = 0$; $T(s, a, s')$ describes the state-to-state transition probabilities (e.g. robot motion model); $O(s, a, z)$ describes the observation probability distribution (e.g. sensor model). And $R(s, a) : S \times A \longrightarrow \Re$ is a (bounded) reward function quantifying the utility of each action for each state.

2.2 Belief Computation

POMDPs assume that the state s_t is not directly observable, but instead the agent perceives observations $\{z_1, \ldots, z_t\}$ which convey information about the state. From these, the agent can compute a *belief*, or probability distribution over possible world states: $b_t(s) = Pr(s_t = s \mid z_t, a_{t-1}, z_{t-1}, \ldots, a_0)$. Because POMDPs are instances of Markov processes, the belief b_t at time t can be calculated recursively, using only the belief one time step earlier, b_{t-1}, along with the most recent action a_{t-1} and observation z_t:

$$b_t(s) = \tau(b_{t-1}, a_{t-1}, z_t) := \frac{\sum_{s'} O(s', a_{t-1}, z_t)\ T(s, a_{t-1}, s')\ b_{t-1}(s')}{Pr(z_t | b_{t-1}, a_{t-1}).} \tag{1}$$

This is equivalent to the Bayes filter, and in robotics, its continuous generalization forms the basis of the well-known Kalman filter. In many large robotics applications, tracking the belief can be computationally challenging. However in POMDPs, the bigger challenge is the generation of an action-selection policy. We assume throughout this paper that the belief can be computed accurately, and focus on the problem of finding good policies.

2.3 Policy Computation

The POMDP framework's primary purpose is to optimize an action-selection *policy*, of the form: $\pi(b) \longrightarrow a$, where b is a belief distribution and a is the action chosen by the policy π. We say that a policy $\pi^*(b_t)$ is optimal when the expected future discounted reward is maximized:

$$\pi^*(b_t) \quad = \quad \operatorname*{argmax}_{\pi} E_\pi \left[\sum_{t=t_0}^{T} \gamma^{t-t_0} r_t \middle| b_t \right]. \tag{2}$$

Computing an optimal policy over all possible beliefs can be challenging [2], and so many recent POMDP approximations have been proposed which gain computational advantage by applying value updates at a few specific belief points [5, 7, 10, 11]. These techniques differ in how they select the belief points, but all use the same procedure for updating the value over a fixed set of points. The key to updating a value function over a fixed set of beliefs, $B = \{b_0, b_1, ..., b_q\}$, is in realizing that the value function contains at most one α-vector for each belief point, thus yielding a fixed-size solution set: $\Gamma_t = \{\alpha_0, \alpha_1, \ldots, \alpha_q\}$.

The standard procedure for point-based value update is the following. First we generate intermediate sets $\Gamma_t^{a,*}$ and $\Gamma_t^{a,z}, \forall a \in A, \forall z \in Z$:

$$\begin{aligned} \Gamma_t^{a,*} &\leftarrow \{\alpha^{a,*}\}, \text{ where } \alpha^{a,*}(s) = R(s,a) && (3)\\ \Gamma_t^{a,z} &\leftarrow \{\alpha_i^{a,z} \mid \alpha_i \in \Gamma_{t-1}\}, \text{ where } \alpha_i^{a,z}(s) = \gamma \sum_{s' \in S} T(s,a,s')O(s',a,z)\alpha_i(s'). \end{aligned}$$

Next, we take the expectation over observations and construct $\Gamma_t^b, \forall b \in B$:

$$\Gamma_t^b \leftarrow \{\alpha^{a,b} \mid a \in A\}, \text{ where } \alpha^{a,b} = \Gamma_t^{a,*} + \sum_{z \in Z} \operatorname*{argmax}_{\alpha \in \Gamma_t^{a,z}} \sum_{s \in S} \alpha(s)b(s). \quad (4)$$

Finally, we find the best action for each belief point:

$$\Gamma_t \leftarrow \{\alpha^b \mid b \in B\}, \text{ where } \alpha^b = \operatorname*{argmax}_{\alpha \in \Gamma_t^b} \sum_s \alpha(s)b(s). \quad (5)$$

Because the size of the solution set Γ_t is constant, the point-based value update can be computed in polynomial time. And while these operations preserve only the best α-vector at each belief point $b \in B$, an estimate of the value function at any belief in the simplex (including $b \notin B$) can be extracted from the set Γ_t:

$$V_t(b) = \max_{\alpha \in \Gamma_t} \sum_{s \in S} \alpha(s)b(s). \quad (6)$$

2.4 Error Bound on Point-Based Value Updates

The point-based value update operation is an integral part of many approximate POMDP solvers. As shown in [5], given a fixed belief set B and planning horizon t, the error over multiple value updates is bounded by[1]:

$$\|V_t^B - V_t^*\|_\infty \leq \frac{(R_{\max} - R_{\min}) \max_{b' \in \Delta} \min_{b \in B} \|b - b'\|_1.}{(1-\gamma)^2}$$

where $b' \in \Delta$ is the point where the point-based update makes its worst error in value update, and $b \in B$ is the closest (1-norm) sampled belief to b'. Now let α be the vector that is maximal at b, and α' be the vector that would be maximal at b'. Then, we can show equivalently that

$$\begin{aligned} \epsilon(b') &\leq \alpha' \cdot b' - \alpha \cdot b' \\ &\leq (\alpha' - \alpha) \cdot (b' - b) \\ &\leq \textstyle\sum_i \begin{cases} (\frac{R_{\max}}{1-\gamma} - \alpha_i)(b'_i - b_i) & b'_i \geq b_i \\ (\frac{R_{\min}}{1-\gamma} - \alpha_i)(b'_i - b_i) & b'_i < b_i. \end{cases} \end{aligned}$$

[1] The error bound proven in [5] depends on the sampling density over the belief simplex Δ. But when the initial belief b_0 is known, it is not necessary to sample all of Δ densely. Instead, we can sample the set of reachable beliefs $\bar{\Delta}$ densely; the error bound holds on $\bar{\Delta}$.

3 Error-Minimization Point Selection

Many recent point-based value approximations, which show good empirical success, use poorly informed heuristics to select belief points. We now describe a new algorithm for selecting provably good belief points. The algorithm directly uses the error bound above to pick those reachable beliefs $b \in \bar{\Delta}$ which most reduce the error bound. Figure 1a shows the tree of reachable beliefs, starting with the initial belief (top node). Building the tree (to a finite depth) is easily done by recursively using Equation 1.

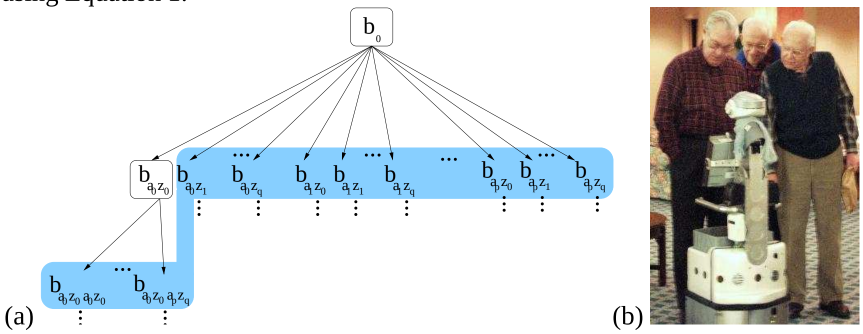

Figure 1: (a) The set of reachable beliefs. Each node corresponds to a specific belief, and increasing depth corresponds to an increasing plan horizon.(b) Pearl the Nursebot interacting with patients in a nursing facility.

Applying point-based value updates to *all* reachable beliefs would guarantee optimal performance, but at the expense of computational tractability: a planning problem of horizon t has $O(|A||Z|^t)$ reachable beliefs. So we select from our reachable beliefs those most likely to minimize the error in our value function. Given the belief tree in Figure 1a, we consider three sets of nodes. Set 1 includes all points already in B (in this example b_0 and b_{a_0,z_0}). Set 2 contains the set of candidates from which we will select new points to be added to B. We call this set the *fringe* (denoted $\bar{B}$). Set 3 contains all other reachable beliefs.[2]

Now we need to decide which belief b should be removed from the fringe $\bar{B}$ and added to the set of active points B. Every new point added to B should improve our estimate of the value function as much as possible. To find the point most likely to do this, we consider the theoretical analysis of Section 2.4. Consider $b' \in \bar{B}$, a belief point candidate, and $b \in B$, some belief which we have already selected. While one could simply pick the candidate $b' \in \bar{B}$ with the largest error bound, $\epsilon(b')$, this would go against the most useful insight from earlier work on point-based approaches: namely that *reachability* considerations are important. So we need to factor in the probability of each candidate belief point occurring. We first note that the error bound at any given belief point b in the tree can be evaluated from that of its immediate descendants:

$$\bar{\epsilon}(b) = \max_{a \in A} \sum_{z \in Z} O(b, a, z) \;\; \epsilon(\tau(b, a, z)) \tag{7}$$

[2] In Figure 1a, the fringe ($\bar{B}$) is restricted to the immediate descendants of the points in B. The rest of the paper proceeds on this assumption, but we could assume a deeper fringe.

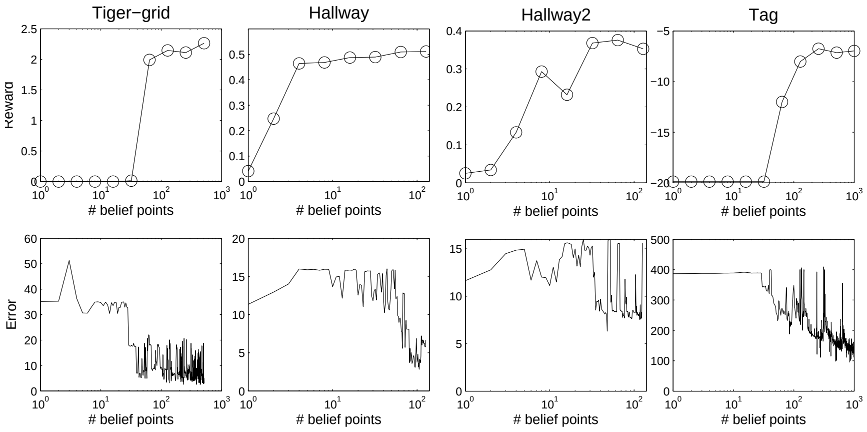

Figure 2: Policy performance (top row) and estimate of the bound on the error (bottom row) for selected belief points

where $\tau(b, a, z)$ is the belief update equation (Eqn 1), and $\epsilon(\tau(b, a, z))$ is evaluated as in Section 2.4 (unless $\tau(b, a, z) \in B$, in which case $\epsilon(\tau(b, a, z)) = 0$). So we use Equation 7 to find the existing point $b \in B$ with the largest error bound, then pick as a new point its descendant $\tau(b, a, z)$ which has the largest impact on $\bar{\epsilon}(b)$. Points on the fringe are picked one a time, allowing us to look deep in the tree; in the experiments presented below, beliefs at 40+ levels are in fact selected.

This concludes the presentation of our new error-minimization point selection technique. In practice, the addition of new points is always interleaved with the point-based value updates described in Section 2.3 to form a full POMDP solution. The complete approach, called PEMA (**P**oint-based **E**rror **M**inimization **A**lgorithm), is now evaluated empirically in a series of robot control experiments.

4 Empirical evaluation

We begin our empirical evaluation with a few well-studied maze navigation domains. Most have been used strictly in simulation, but feature robot-like assumptions, such as non-deterministic motion and noisy sensors. The Tiger-grid, Hallway and Hallway2 problems are described in [3]. The Tag domain was introduced in [5]. The goal of these preliminary experiments is simply to compare the performance of PEMA with earlier POMDP approximations on standard problems. More extensive robot navigation domains are presented in the following section.

Error estimates. A first set of results on PEMA's performance are shown in Figure 2. For each problem domain, we first plot PEMA's reward performance as a function of the number of belief points (top graphs), and then plot the error estimate of each point selected according to the order in which points were picked (bottom graphs). As shown in these, PEMA is able to solve all four problems with relatively few beliefs (sometimes fewer than the number of states).

Considering the error bound graphs, we see that overall there seems to be reasonably good correspondence between an improvement in performance, and a decrease in the error estimates. We can conclude from these plots that the error bound used by PEMA is quite informative in guiding exploration of the belief simplex.[3]

Comparative analysis. While the results outlined above show that PEMA is able to handle a wide spectrum of large-scale POMDP domains, it is also useful to compare its performance to that of alternative approaches, on the same set of problems. Figure 3 compares both reward performance and policy size[4] (# of nodes in controller) for a few recent POMDP algorithms, on the three larger problems (Hallway, Hallway2, and Tag). The algorithms included in this comparison were selected simply based on the availability of published results for this set of problems.

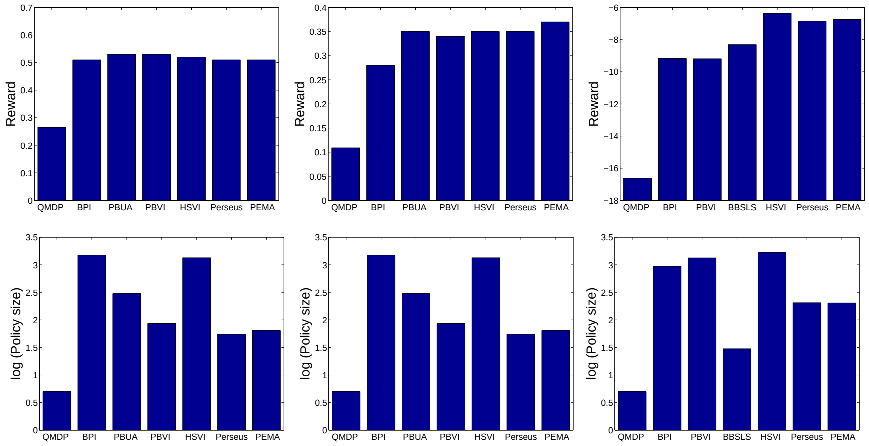

Figure 3: Results for standard POMDP domains. Left column: Hallway problem. Middle column: Hallway2 problem. Right column: Tag problem.

As is often the case, these results show that there is not a single algorithm that is best for solving all problems, so it is difficult to draw broad generalizations. But we can point out a few salient effects. First, the baseline QMDP [3] approximation is clearly outclassed by other more sophisticated methods. We also observe that some of the algorithms achieve sub-par performance in terms of expected reward: BPI [9] (on Hallway2 and Tag)[5], PBVI [5] (on Tag) and BBSLS [1] (on Tag). While each of these is theoretically able to reach optimal performance, they would require larger controllers (and therefore longer computation time) to do so.

The remaining algorithms—HSVI [10], Perseus [11], and PEMA—offer com-

[3] While the decrease in error over a fixed point (e.g. b_0) is monotonic, the decrease in error over *each new points* (in the order it was added) is not necessarily monotonic, which explains the large jumps in the bottom graphs. These jumps suggest that PEMA could be improved by maintaining a deeper fringe of candidate belief points, in which case the time spent selecting points would have to be carefully balanced with the time spent planning. Currently, we spend less than 1% of computation time selecting belief points; the rest is spent estimating the value function.

[4] The results were computed on different platforms, so time comparisons are difficult. The size of the final policy is often a useful indicator of computation time, but should be considered with care.

[5] Better results for BPI have since been published in [8].

parable performance. HSVI offers good control performance on the full range of tasks, but requires bigger controllers. HSVI and PEMA share many similarities: both use an error bound to select belief points. HSVI's upper-bound is tighter than PEMA's, but requires costly LP solutions. PEMA solves problems with fewer belief points, we believe this is because it updates all belief points more frequently, thus generalizing better in poorly explored areas of the belief simplex.

Between Perseus and PEMA, the trade-offs are less clear: the planning time, controller size and performance quality are quite comparable. These two approaches in fact share many similarities. Perseus uses the same point-based backups as in PEMA (see Section 2.3), but it differs in both how the set of belief points is selected (Perseus uses random exploration traces), and the order in which it updates the value at those points (also randomized). The effect of these differences is hard to narrow. We did experiment informally with Perseus-type random updates within PEMA, but this did not yield significant speed-up. It is likely that randomizing value updates is not as beneficial when carefully picking a small set of essential points. We speculate that PEMA will scale better to higher dimensions because of the selective nature of the belief sampling. This is the subject of ongoing work.

5 Robotic applications

Much of the algorithmic development described in this paper is motivated by our need for high-quality robust planning for interactive mobile robots. In particular, we are concerned with the problem of controlling a nursing assistant robot. This is an important technical challenge arising from the Nursebot project [6]. This project aims to develop personalized robotic technology that can improve the level of personal care and services for elderly individuals. The robot Pearl (Fig. 1b) is the main experimental platform used in this project. It is equipped with standard indoor navigation abilities and is programmed with the CARMEN toolkit [4]. An important task for this robot is to provide timely cognitive reminders (e. g. medications to take, appointments to attend, etc.) to its target population. It is therefore crucial that the robot be able to find the person whenever it is time to issue a reminder. We model this task as a POMDP, and use PEMA to optimize a strategy with which the robot can robustly find the person, even under very weak assumptions over the person's initial location and ease of mobility.

We begin by considering the environment in which the robot operates. Figure 4 shows a 2D robot-generated map of its physical environment. The goal is for the robot to navigate in this environment until it finds the patient and then deliver the appropriate reminder. To successfully find the patient, the robot needs to systematically explore the environment, while reasoning about both its spatial coverage and the likely motion pattern of the person.

5.1 POMDP Modeling

To model this task as a POMDP, we assume a state space consisting of two features: *RobotPosition*, and *PersonPosition*. Each feature is expressed through a fixed discretization of the environment (roughly 25 cells for each feature, or 625 total states.) We assume the person and robot move freely, constrained only by walls and obstacles. The robot's motion is deterministic (as a function of the action={*North, South, East, West*}). A fifth action (*DeliverMessage*) concludes the scenario if applied when the robot and person are in the same location. We assume the person's motion is stochastic, and in one of two modes: (1) whenever the person is far from the robot, s/he moves according to Brownian motion (i. e. in each cardinal direction with $Pr = 0.1$ or stays in place), this corresponds to a random walk and is a conservative assumption regarding people's motion; or (2) whenever the robot is within sight ($<$ 4m), the person tries to avoid the robot and moves away from it (with noise), which makes the task more challenging.

The observation function has two parts: what the robot senses about its own position, and what it senses about the person's position. First we assume that the robot's position is fully known; this is reasonable since planning is done at a much coarser resolution (2m), than the typical localization precision (10cm). When testing policies however, probabilistic localization is performed by the CARMEN toolkit, and the robot's belief incorporates any positional uncertainty. For the person's position, we assume that the robot perceives nothing unless the person is within 2 meters. This is plausible given the robot's sensors. Even at short-range, there is a small probability ($Pr = 0.01$) that the robot will miss the person.

The reward function is straightforward: $R = -1$ for any motion, $R = 10$ when the robot decides to *DeliverMessage* and is within range (<2m) of the person, and $R = -100$ when the robot decides to *DeliverMessage* in the person's absence. The task terminates when the robot successfully delivers the message. We assume a discount factor proportional to the map's resolution ($\gamma = 0.98$).

With these POMDP parameters, we can run PEMA to optimize the robot's control strategy. Given the complexity of POMDP planning we do assume that PEMA will be used as an off-line algorithm to optimize the robot's performance prior to deployment. The results presented below describe the performance of an optimized control policy when tested onboard the CARMEN simulator.

5.2 Experimental Results

We first consider PEMA's performance on this task, as a function of planning time. As shown in Figure 4a, PEMA is in fact able to solve the problem within 1800 seconds, using only 128 belief points. In comparison, an MDP-type approximation (in this case the QMDP technique [3]) proves to be inadequate for a problem exhibiting such complex uncertainty over the person's position. Using PEMA, the patient was found in 100% of trials, compared to 35% for QMDP.

Figure 4 shows PEMA's policy through five snapshots from one run. The policy is optimized for any start positions (for both the person and the robot); the execution trace in Figure 4 is one of the longer ones since the robot searches the entire environment before finding the person. In this scenario, the person starts at the far end of the left corridor. The person's location is not shown in the figure since it is not observable by the robot. The figure instead shows the *belief* over person positions, represented by a distribution of point samples (grey dots). We see the robot starting at the far right end of the corridor (Fig. 4b), moving towards the left until the room's entrance (Fig. 4c), and searching the entire room (Fig. 4d). Once sufficiently certain that the person is not there, it exits the room (Fig. 4e), and moves towards the left until it finally finds the person at the end of the corridor (Fig. 4f).

It is interesting to compare snapshots (b) and (d). The robot position in both is practically identical. Yet in (b) the robot chooses to go up into the room, whereas in (d) the robot chooses to move toward the left. This is a direct result of planning over *beliefs*, rather than over *states*.

These results show that PEMA is able to handle realistic domains. In particular, throughout these experiments, the robot simulator was in no way constrained to behave as described in our POMDP model. For example the robot's actions often had stochastic effects, the robot's position was not always fully observable, and belief tracking had to be performed asynchronously (i. e. not a straight alternation of actions and observations). Despite this mismatch between the model assumed for planning and the execution environment, the control policy optimized by PEMA successfully completed the task.

5.3 Robustness to modeling errors

Like most POMDP solvers, PEMA assumes exact knowledge of the POMDP model. In reality, this model is often hand-crafted and may bear substantial error. In our experience, such a mismatch between model and the real system does not necessarily render our solution useless. The robustness built in to POMDPs to overcome state uncertainty often goes a long way towards overcoming model uncertainty. Nonetheless, there are cases where a poor model can be catastrophic. In this section, we try to gain a better understanding of the impact of errors in the model we used for the Find-the-patient domain.

Our model assumes that the robot can see the patient with $Pr = 0.99$, whenever s/he is within 2m. We use this parameter both for solving and tracking. But it could be that in fact the person is only detected with $Pr = 0.8$.

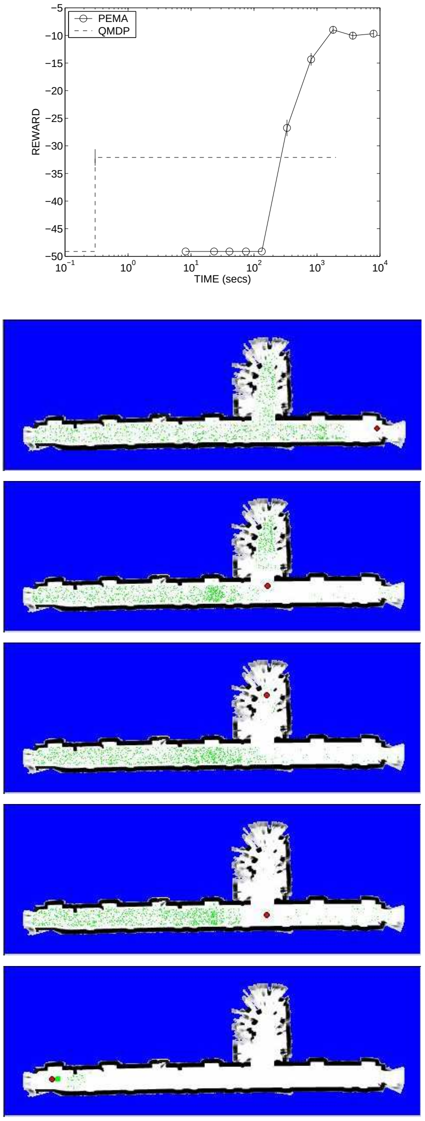

Figure 4: Find-the patient domain: (a) Performance results. (b)-(f) Sample trajectory.

$\Pr_{model}(z)$	$\Pr_{real}(z)$ 0.99	 0.90	 0.80	 0.70
0.99	**-9.7**	-11.3	-13.2	-15.5
0.90	-12.0	**-13.1**	-15.6	-19.0
0.80	-9.7	-11.5	**-13.1**	-14.5
0.70	-17.8	-19.4	-22.0	**-22.6**

Table 1: Sensitivity analysis over observation probabilities. (CI for all: [0.7,1.4])

What would be the loss in performance, compared to if we had planned and tracked with the correct parameter? Table 1 examines the effects of this type of modeling error. It shows the performance (avg. sum of rewards over 1000 trajectories) when applying PEMA and tracking the belief with the sensor accuracy in the left column, but testing with the accuracy in the top row. The main diagonal contains cases where the model is correct. These results suggest two things. First, as expected, performance degrades as the real noise level increases (i.e. left-to right effect for any given row.) Second, and this was not anticipated, the dominating performance factor is in fact the noise in the assumed model: regardless of what conditions are used for testing, results are better for some values of Pr_{model} $(0.99$ and $0.8)$ and worse for others $(0.9$ and $0.7)$. We hypothesize that this happens because in some models, PEMA did not have sufficient belief points to perform well (all policies were optimized with $|B|$=512). When we repeated experiments for $\text{Pr}_{model}(z)$=0.9 with more beliefs points, the performance improved (for all $\text{Pr}_{real}(z)$) to the level of the top row. This suggest that in some domains it may be best to optimize policies assuming false models (e. g. low sensor noise), because an equally good policy can be obtained with fewer belief points. We are currently investigating this, as well as the impact of modeling errors in the transition model.

6 Conclusion

This paper describes a new algorithm for planning in partially observable domains, which features a theoretically-motivated technique for selecting salient information states. This improves the scalability of the approach, to the point where it can be used to control a robot seeking a missing person. We also demonstrate that the algorithm is robust to noise in the assumed model. Future work focuses on improving performance under even weaker modeling assumptions.

References

[1] D. Braziunas and C. Boutilier. Stochastic local search for POMDP controllers. In *AAAI*, 2004.

[2] A. Cassandra, M. L. Littman, and N. L. Zhang. Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 54–61, 1997.

[3] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In *Proceedings of Twelfth International Conference on Machine Learning*, pages 362–370, 1995.

[4] M. Montemerlo, N. Roy, and S. Thrun. Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit. In *Proceedings of IROS*, 2003.

[5] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of IJCAI*, 2003.

[6] J. Pineau, M. Montermerlo, M. Pollack, N. Roy, and S. Thrun. Towards robotic assistants in nursing homes: challenges and results. *Robotics and Autonomous Systems*, 42(3-4), 2003.

[7] K.-M. Poon. A fast heuristic algorithm for decision-theoretic planning. Master's thesis, Hong-Kong Univ. of Science and Technology, 2001.

[8] P. Poupart. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. PhD thesis, University of Toronto, 2005.

[9] P. Poupart and C. Boutilier. Bounded finite state controllers. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, 2004.

[10] T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *Proc. of UAI*, 2004.

[11] N. Vlassis and M. T. J. Spaan. A fast point-based algorithm for POMDPs. In *Belgian-Dutch Conference on Machine Learning*, 2004.