Approximate Planning with Hierarchical Partially Observable Markov Decision Process Models for Robot Navigation

Georgios Theocharous
theochar@cse.msu.edu
Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48823

Sridhar Mahadevan
mahadeva@cs.umass.edu
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

Abstract— **We propose and investigate a planning framework based on the Hierarchical Partially Observable Markov Decision Process model (HPOMDP), and apply it to robot navigation. We show how this framework can be used to produce more robust plans as compared to flat models such as Partially Observable Markov Decision Processes (POMDPs). In our approach the environment is modeled at different levels of resolution, where abstract states represent both spatial and temporal abstraction. We test our hierarchical POMDP approach using a large simulated and real navigation environment. The results show that the robot is more successful in navigating to goals starting with no positional knowledge (uniform initial belief state distribution) using the hierarchical POMDP framework as compared to the flat POMDP approach. In addition, the HPOMDP model allows the robot to efficiently model and navigate large scale environments.**

I. Introduction

One of the most difficult challenges in programming autonomous mobile robots is to enable them to act reliably in large, and perceptually aliased environments (e.g., in an office environment, two corridors or intersections can look alike). Animals use path integration and other long-term memory structures to solve this problem. Humans can easily follow directions that require remembering past decisions (e.g., take a right at the first stop sign, and then a left at the next stop sign). Additionally, to handle large scale environments, humans and animals seem to use multi-resolution representations. For example, in driving from East Lansing, Michigan to Amherst, Massachusetts we use both local city maps as well as interstate maps. An abstract representation of a navigation task not only reduces the size of the representation (state space) but also reduces perceptual aliasing. For example, once we are on the highway we don't care where exactly we are, but rather follow the road until we see the desired exit sign.

An autonomous mobile robot that has to navigate in an indoor office environment needs to know where it is at each point in time so that it can make appropriate decisions (e.g., at this T-junction, should I turn left or right?). Unfortunately indoor office environments are perceptually aliased. Even if the robot had perfect sensors, which is usually not the case, it will not be able to distinguish in which place it is in its environment simply by observing its surroundings since many locations look alike. One way to know exactly where the robot is in the environment is to keep track of odometry. Unfortunately, this also requires noise-free dead-reckoning and knowledge of the starting position of the robot. One solution to the above problems is to model the navigation task as a partially observable Markov decision process (POMDP) [1], [2],[3]. In a POMDP representation of robot navigation, states correspond to fixed length regions, e.g. each square meter in the environment is modeled as four states of the POMDP (one for each robot orientation). The robot can take actions such as "go-forward", "turn-left", and "turn-right". In each state, the robot can observe features such as walls, openings, and doors on the front, left, back and right side. Unfortunately, modeling a real office environment in this manner requires thousands of states and exact POMDP planning methods based on dynamic programming are intractable. Also, heuristic planning methods (e.g. Q-MDP or MLS) are not practical since they are usually effective in belief states with unimodal low entropy distributions [1].

In this paper we investigate a multi-resolution model called Hierarchical Partially Observable Markov Decision Processes (HPOMDPs), which was recently proposed in [4]. We describe a general hierarchical planning algorithm using HPOMDPs and apply the framework to corridor navigation to produce and execute hierarchical plans. At an abstract level we find mappings from abstract states (e.g whole corridors) to macro-actions (e.g, go down the corridor) and at lower resolutions we produce local plans, such as "exit a corridor" or reach a goal location within a corridor. Our experiments show that there is less uncertainty associated with the belief states defined over higher levels of the HPOMDP hierarchy. As a result, heuristic POMDP solutions allow the robot to reach selected destinations of its environment starting from a state of no knowledge of its original starting location. In addition, the hierarchical nature of our model allows us to represent larger environments more efficiently since we don't have to represent all relationships between states in the environment at a uniform resolution. Also, hierarchical plans are computed much faster since they can be constructed from precomputed local plans. In the rest of the paper, we describe the Hierarchical Hidden Markov model (HHMM), its extension to HPOMDP, how it can be used for planning, its application to robot navigation, and experiments using large simulated and real environments.

II. Hierarchical Hidden Markov Models

The HPOMDP model was derived from the HHMM [5] model with the addition of primitive and abstract actions, and reward functions. The HHMM generalizes the standard hidden Markov model (HMM) [6], by allowing hidden states to represent stochastic processes themselves. An HHMM is visualized as a tree structure (see Figure 1) in which there are three types of states, production states (leaves of the tree) which emit observations, and internal states which are (unobservable) hidden states that represent entire stochastic processes. Each production state is associated with an observation vector which maintains distribution functions for each observation defined for the model. Each internal state is associated with a horizontal transition matrix, and a vertical transition vector. The horizontal transition matrix of an internal state defines the transition probabilities among its children. The vertical transition vectors define the probability of an internal state to activate any of its children. Each internal state is also associated with a child called an *end-state* which returns control to its parent. The end-states do not produce observations and cannot be activated through a vertical transition from their parent.

Figure 1 shows a graphical representation of an example HHMM. The HHMM produces observations as follows:

1. If the current node is the root, then it chooses to activate one of its children according to the vertical transition vector from the root to its children.
2. If the child activated is a product state, it produces an observation according to an observation probability output vector. It then transitions to another state within the same level. If the state reached after the transition is the end-state, then control is returned to the parent of the end-state.
3. If the child is an abstract state then it chooses to activate one of its children. The abstract state waits until control is returned to it from its child end-state. Then it transitions to another state within the same level. If the resulting transition is to the end-state then control is returned to the parent of the abstract state.

III. Hierarchical Partially Observable Markov Decision Process Models

An extension to the HHMM model which includes actions and rewards was proposed in [4], which also showed how such models could be learned from sequences of observations. Here we extend the definition to include multiple entry points into abstract states which we call *entry-states*, and multiple *exit-states* (previously defined as end-state). Figure 3 shows a sample HPOMDP with multiple entry/exit states for corridor environments. The exit and entry states represent the spatial borders of each corridor abstract state. If we only had single exit and entry states, the consequences of primitive actions would not be modeled correctly. For example, an exit from the east side of a corridor would have a non-zero transition probability to an adjacent state of the corridor at the west side. On the other hand, we could have two abstract states representing each corridor (one for each direction). However, having two abstract states for the same spatial locations would mean that the abstract states share children which is not allowed by our model (since it would violate our concept of spatial abstraction). We formally describe the elements of an HPOMDP below:

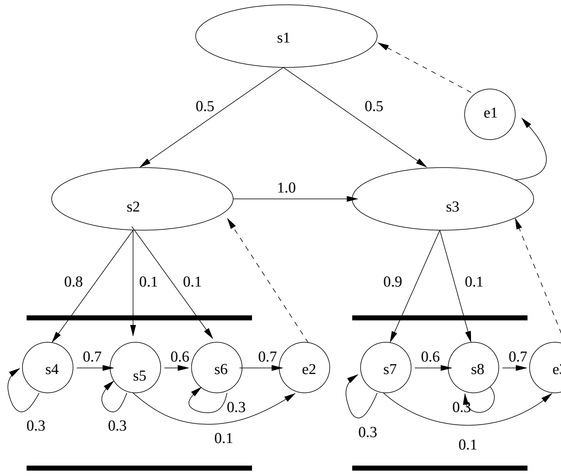

Fig. 1. An example hierarchical hidden Markov model (HHMM) which represents two adjacent corridors. Only leaf (production) states (s4, s5, s6, s7, and s8) produce observations. s2 and s3 represent corridors.

- S denotes the set of states. Unlike HHMMs, we have an additional type of state called entry states.
  - *Product states* which produce observations.
  - *Exit-states* which return control to their parent abstract state when entered.
  - *Entry-states* which belong to abstract states and when entered activate the children of the abstract state they are associated with.
  - *Abstract states* which group together *product*, *entry*, *exit*, and other *abstract* states. We say C^s to denote the product children, and entry states of other abstract states which are children of abstract state s. We say X^s to denote the set of exit states that belong to s and N^s to denote the set of entry-states that belong to abstract state s.
- A denotes the set of primitive actions. For example, in robot navigation the actions could be *go-forward* one meter, *turn-left* 90 degrees, and *turn-right* 90 degrees. Primitive actions are only taken at product states.
- $T(s'|s,a)$ denotes the horizontal transition probabilities, for primitive actions on *product* states, where s is the current product state and s' is the next state which can be either an *entry-state* or a *product-state*.
- $TE(s'|s_x,a)$ denotes the horizontal transition probabilities, for abstract states, where s is the current abstract state, and x is *the exit-state* that abstract state s was ex-

ited from under primitive action a, s' is the next state which can be either an *entry-state* or a *product-state*.
• $V(i|s_n)$ denotes the probability that entry state n which belongs to abstract state s will activate child state i. Child i could be a product state or an entry-state of some abstract state (other than s).
• Z denote the set of observations.
• $P(z|s,a)$ denotes the probability of observation z in product state s after action a has been taken.
• $R(s,a)$ denotes an immediate reward function defined on the product states.

IV. Robot Navigation in Corridor Environments

Figure 2 shows a section of the physical environment at the MSU engineering building. An immediate observation is that corridors are quite long (more than 1000 ft.) and locations within the corridors do not have distinguishing characteristics for a robot to localize. However, this problem can be alleviated if we treat whole corridors as single states and thus abstract away information that may be unnecessary for most navigation tasks. The reason is that for most navigation tasks a robot just needs to know in which corridor it is in, rather than the precise corridor location, and therefore it can often ignore location uncertainty that arises within corridors.

Fig. 2. The corridor on the left is a section of the MSU engineering building. Pavlov, the robot on the right, uses 16 sonar sensors to navigate the whole engineering building.

Figure 3 shows an HPOMDP model for robot navigation and figure 4 shows the equivalent flat model. Both models are used to represent corridor environments. For every two meters there are four product states (one for each direction). In the HPOMDP model, whole corridors are represented with a single abstract state, which in addition to the product states contains two entry-states and two exit-states for the entry and exit points of the corridor. Junction nodes are represented with four product states as shown in Figure 3. In the HPOMDP model for corridor environments, we use the term product level to denote the set of all product states, and abstract level denotes all abstract states plus the product states that belong to junctions.

A multi-resolution representation is intuitive for many reasons. First of all it allows us to scale up to larger environment since we do not need to represent all relationships

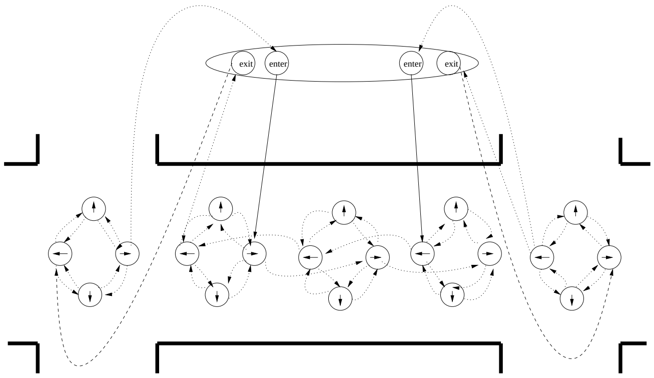

Fig. 3. This HPOMDP is used to model corridor environments. The dotted arrows from product states show non-zero transition probabilities for primitive actions. The black vertical arrows show non-zero vertical activations of product states from the entry states. The dashed arrows from exit states show non-zero transition probabilities from exit-states to adjacent states of the abstract state they are associated with.

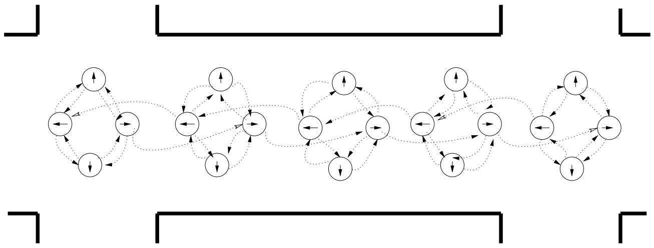

Fig. 4. This is the equivalent flat POMDP of the Hierarchical POMDP shown in Figure 3. In this model we only represent the product states of the hierarchical model. Global transition matrices for each action are automatically computed from the hierarchical model by considering all the possible paths between the product states under the primitive action.

among product states. Second, for many navigation tasks the robot does not need to know where it is in a corridor but rather that it is in some corridor which it has to traverse. Third, it offers us the capability of learning local policies for the different abstract states which we can reuse for any global plan at the abstract level. And finally, as shown in Figure 5, the robot's belief as to where it is at the abstract level is less uncertain from its belief as to where it is at a global product level. In fact, the entropy plot reveals a number of characteristics that motivate our planning algorithm:
1. It is obvious from the plot that uncertainty is less at an abstract level.
2. Uncertainty reduces dramatically at junctions both at the global and abstract levels.
3. At the global level uncertainty increases as the robot traverses corridors where all perceptions are usually the same, but at an abstract level it does not increase.
4. Overall uncertainty decreases as the robot traverses a series of corridors.

The above characteristics suggest that a good approximate algorithm could be one that treats the problem at the abstract level as completely observable. In addition, a successful algorithm would be one that always takes the

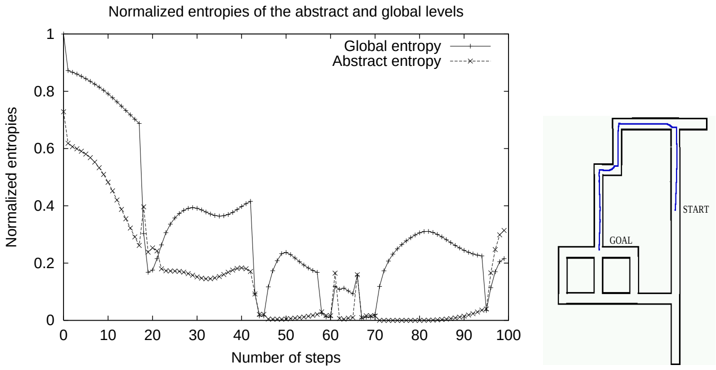

Fig. 5. The plot on the left shows the normalized entropies of the belief of the robot location at the global product level and at the abstract level. The entropy plot was created during a robot run on the Nomad 200 simulator whose trace is shown on the right.

robot from every entry-state of a corridor to the exit-state at the other exit point of the corridor so that overall uncertainty is gradually reduced. In the next section we formally present such an algorithm which is based on the theory of Markov decision processes (MDPs).

V. Planning and Execution Algorithm

A. Planning

To find a completely observable plan at an abstract level we treat the product states and the entry states (of the abstract states) at an abstract level as the states of an MDP. The set of actions is the set of primitive actions A applicable on product states and the set of macro-actions for each entry state n of every abstract state s, which we define to be M^{s_n}, and are responsible for taking the agent out of the abstract state s, or to some child state of the abstract state. For our corridor environment we defined at the most two macro-actions to be available at each entry state, the one that will take the robot to the opposite exit-state and an additional macro action if the goal belongs to the same abstract state as the entry state. The reason is that we always want the robot to traverse corridors such that the uncertainty is gradually reduced. We can design macro-actions for each corridor by either hard-coding them or by computing them using the MDP framework where we only give a goal reward. Figure 6 shows different macro-actions for corridors.

Given these macro-actions we need to evaluate them for each entry state s_n. Equation 1 gives us the transition probability from entry-state $s_n \in N^s$ to an adjacent state s' of the abstract state s under some macro-action μ. An exit from state s can occur from any exit state x under the primitive action $\pi^s_\mu(s_x)$, where π^s_μ defines the policy of macro action μ on all states i that are either in the C^s or X^s sets. If i is a product state then $\pi^s_\mu(i)$ is a primitive action. If i is an entry state then $\pi^s_\mu(i)$ is a macro action that belongs to $M^{s'_i}$, where s' is the abstract state associated with entry state i.

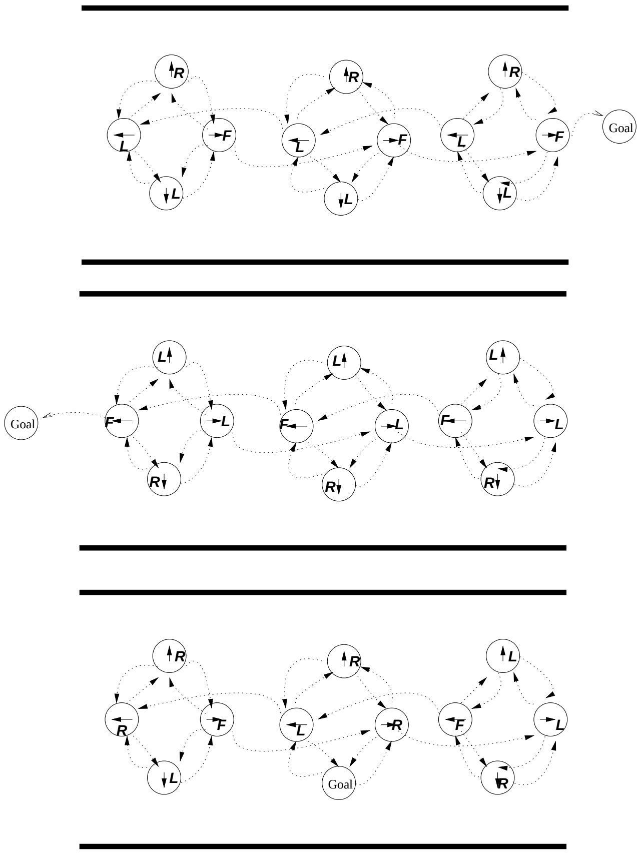

Fig. 6. F means go forward, R means turn right and L means turn left. The top figure is a policy for exiting the East side of the corridor with the go-forward action, the middle figure is a policy for exiting the West side of the corridor with the go-forward action, and the bottom figure is a policy for reaching a particular location within the corridor.

$$T(s'|s_n,\mu) = \sum_{\forall s_x \in X^s} \left[\sum_{\forall i \in C^s} V(i|s_n) TD(s_x|i,\mu) \right] TE(s'|s_x, \pi^s_\mu(s_x)) \tag{1}$$

The term $TD(x|i,\mu)$ is the expected discounted transition probability for exiting the parent of state i starting from i and can be calculated by solving a set of linear equations defined in Equation 2.

$$\begin{aligned} TD(s_x|i,\mu) =& T(s_x|i,\pi^s_\mu(i)) + \\ & \gamma \sum_{j \in C^s} T(j|i,\pi^s_\mu(i)) TD(s_x|j,\mu) \end{aligned} \tag{2}$$

In a similar manner we can calculate the reward R that will be gained when an abstract action μ is executed at an entry state s_n as shown in Equation 3.

$$R(s_n,\mu) = \sum_{i \in C^s} V(i|s_n) RD(i,\mu) \tag{3}$$

where the term $RD(i,\mu)$ can be calculated by solving a set of linear equations defined in Equation 4.

$$RD(i,\mu) = R(i,\pi_\mu^s(i)) + \gamma \sum_{j \in C^s} T(j|i,\pi_\mu^s(i)) RD(j,\mu) \tag{4}$$

Given any arbitrary HPOMDP and all the macro-actions M^{s_n} of every entry state of every abstract state we can calculate all the reward and transition models at the highest level by starting from the lowest levels and and moving toward the top. Discussion of macro-action generation and usage can be found in [7] and also in the more general theory of the *options* framework [8]. The major differences in this paper is that first of all applicability of each macro-action is defined by the abstract states and exit-states which can be though of as special cases of initiation and termination conditions in the *options* framework. And second, when it comes to the construction of global plans we compute them over a selected number of states (states at the abstract level).

Once the transition and reward models are calculated we can construct an abstract plan among product and entry-states at the abstract level. To find a plan, we first compute the optimal values of states by solving a set of linear equations defined by the Bellman equation (as shown below).

$$V^*(s_1) = \max_{a \in A | a \in M^{s_1}} (R(s_1,a) + \gamma \sum_{s_2}^{C^{root}} T(s_2|s_1,a) V^*(s_2)) \tag{5}$$

where a is either a primitive action or a macro action. We can then associate the best action for each state (or optimal policy π^*) greedily as shown in Equation 6.

$$\pi^*(s_1) = \text{argmax}_{a \in A | a \in M^{s_1}} (R(s_1,a) + \gamma \sum_{s_2}^{C^{root}} T(s_2|s_1,a) V^*(s_2)) \tag{6}$$

B. Execution

To execute the abstract plan we keep track of an "abstract" probability distribution over states at the abstract level. An efficient way to calculate the abstract belief state is to first calculate a global transition matrix among all product states under the primitive actions, and use that transition matrix to update a global belief state among product states after every action and observation as shown in Equation 7.

$$b'(s) = \frac{1}{P(z|a,b)} P(z|s,a) \sum_{s'=1}^{|S|} P(s|a,s') b(s') \tag{7}$$

where b' is the next belief state, $P(z|s,a)$ is the probability of perceiving observation z when action a was taken in state s. In the navigation case, we have 16 observations which indicate the probability of a wall and opening on the four sides of the robot. The observations are extracted from trained neural nets where the inputs are local occupancy grids constructed from sonar sensors and outputs are probabilities of walls and openings [9]. $P(s|a,s')$ is the probability of going from state s' to state s under action a and b is the previous belief state. Using the flat belief state b we can calculate the abstract belief state using Equation 8.

$$\begin{aligned} B(s) &= b(s) \;\; \text{if s is a product state} \\ B(s) &= \sum_{c(s,i)}^{C^s} b(c(s,i)) \text{ if s is an abstract state} \end{aligned} \tag{8}$$

After every action and observation we update the abstract belief state. If the most likely state s is a product state, then we execute the action associated with s. If the most likely state s is an abstract state then we find the most likely entry state s_n by summing the beliefs of the children of the abstract state that have a non-zero vertical transition from the entry state s_n. We then execute the appropriate macro-action until abstract state s is no longer the most likely state. There are different ways to execute the macro-action based on heuristic POMDP solutions that assume that the underlying model is completely observable [1]. One way is to use the most likely state heuristic where we initialize the local probability distribution over the children of s according the vertical activation from entry-state s_n.

VI. Experimental Results

We applied the above algorithm to a large scale robot navigation environments shown in Figure 7. For the small environment the total number of states is 457 for POMDPs and 532 for HPOMDPs. For the large environment the total number of states is 1285 for POMDPs and 1385 for HPOMDPs. In both environments the task of the robot was to get to the four-way starting from a uniform initial global belief state. To formulate this as a goal planning problem we gave the robot a -1 reward for every action executed except for $+1$ reward for actions leading to the goal state. In this way the robot would learn to choose shorter routes so as to minimize the cost. For the simulated environments we tested the algorithm by starting the robot from 30 different random locations and observed the number of steps it took to reach the goal. We also ran the robot in the real environment starting it from 23 locations and using the HPOMDP model. We did not use the flat POMDP model in the real environment due to its large size, long planning times, and the limited computing power on the real robot. Table I shows the results for the different environments and the models used. It is clear that the HPOMDP models give 100 % success in simulation and high success rate in the real world. Unsuccessful trials in the real world were due to failures of the behavior-based layer of the navigation architecture that was responsible for executing the primitive actions (e.g., crash into a wall).

We also measured the time it took for the robot to construct a plan for the different environments and the differ-

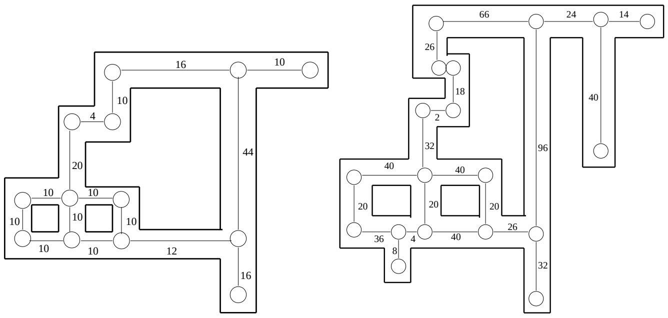

Fig. 7. The figure shows a small environment on the left that was used with the Nomad 200 simulator only and a large environment on the right that was used both in simulation and in the real world. The number next to the edges is the distance between the nodes in meters.

Envir.	Model	Goal %	Steps	Abstract Decisions
small sim.	HPOMDP	100	87.1	19.8
small sim.	POMDP	40	96.2	96.2
large sim.	HPOMDP	100	176.9	20.2
large sim.	POMDP	50	184	184
large real	HPOMDP	86.6	176.5	23.5

TABLE I

The table shows percentage of successful trials where the goal was reached, the average number of primitive steps, and the average number of decision at the abstract level starting from an initial maximum entropy belief state (uniform distribution).

ent models. The results are summarized in table II where for the HPOMDP model it takes significantly less time to compute a plan.

Environment	Model	Planning Time (sec)
small	HPOMDP	4.22
small	POMDP	25.05
large	HPOMDP	38.16
large	POMDP	409.92

TABLE II

The table shows the planning times for the different environments and models.

VII. Conclusions and Future Directions

Our hierarchical approach to robot navigation using the HPOMDP model seems to outperform previous flat POMDP based representations. The algorithm presented is highly successful in taking the robot to the goal starting from no positional knowledge. Also, the hierarchical approach allows us to scale up to larger domains both in terms of time and space. Abstract plans are constructed much faster than flat plan representations, and the amount of space needed to represent the model is a lot less since not all relationships between product states are represented.

The high success rate in taking the robot to its goal is due to two key reasons. First, the robot takes decisions at an abstract level where there are less number of states to consider and as a result less uncertainty. And second, when the robot enters a corridor it chooses the macro-action to move to the opposite exit of the corridor which results in reduction of the entropy. Even though going to the opposite end of the corridor may not be optimal if the environment was completely observable, in POMDPs it makes sense because it serves as an information gathering action for places where uncertainty is high (which we know to be inside long corridors).

In addition to the fact that this algorithm is successful for robot navigation in corridor environments it may have application to domains other than robotics. Currently, it is up to the designer to group together high entropy states. One future direction would be to infer the structure of the model as well, where the goal would be to group together states where uncertainty rises. Another future direction would be to apply the model and algorithm to deeper hierarchies with more than 2 levels. And finally, a future direction would be the study of this model with respect to exact POMDP solutions. A hierarchical approach for planning and execution in POMDPs that is closer to exact POMDP solutions is described in [10].

Acknowledgments

This research is supported in part by a DARPA MARS grant DABT63-99-1-0014, and an NSF Knowledge and Distributed Intelligence (KDI) Award ECS-9873531.

References

[1] Sven Koenig and Reid Simmons, "A Robot Navigation Architecture Based on Partially Observable Markov Decision Process Models," in *Artificial Intelligence Based Mobile Robotics:Case Studies of Successful Robot Systems*, D. Kortenkamp, R. Bonasso, and R. Murphy, Eds., pp. 91–122. MIT press, 1998.

[2] Hagit Shatkay and Leslie Kaelbling, "Learning Topological Maps with Weak Local Odeometric Information," in *Proc. of the International Joint Conference on Artificial Intelligence*, 1997.

[3] I. Nourbakhsh, R. Powers, and S. Birchfield, "Dervish: An office-navigation robot," in *AI Magazine 16(2):53-60.* 1995.

[4] Georgios Theocharous, Khashayar Rohanimanesh, and Sridhar Mahadevan, "Learning hierarchical partially observable markov decision processes for robot navigation," in *IEEE Conference on Robotics and Automation (ICRA)*, Seoul, Korea, 2001.

[5] Shai Fine, Yoram Singer, and Naftali Tishby, "The Hierarchical Hidden Markov Model: Analysis and Applications," *Machine Learning*, vol. 32, no. 1, pp. 41–62, July 1998.

[6] Lawrence Rabiner, "Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," in *Proceedings of the IEEE*, February 1989, vol. 77.

[7] Milos Hauskrecht, Nicolas Meuleau, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier, "Hierachical solution of Markov decision processes using macro-actions," in *UAI-98*, 1998.

[8] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, pp. 112:181–211, 1999.

[9] Sridhar Mahadevan, Georgios Theocharous, and Nikfar Khaleeli, "Fast concept learning for mobile robots," in *Autonomous Robots Journal, vol. 5, pp. 239-251.* 1998.

[10] J. Pineau, N. Roy, and S. Thrun, "A hierarchical approach to POMDP planning and execution," in *Workshop on Hierarchy and Memory in Reinforcement Learning (ICML 2001)*, Williams College, MA, June 2001.