To appear in: Proc. of IEEE/RSJ International Conf. on Intelligent Robots and Systems, 2006

# Planning and Acting in Uncertain Environments using Probabilistic Inference

Deepak Verma
Dept. of CSE, Univ. of Washington,
Seattle, WA 98195-2350
deepak@cs.washington.edu

Rajesh P. N. Rao
Dept. of CSE, Univ. of Washington,
Seattle, WA 98195-2350
rao@cs.washington.edu

***Abstract*— An important problem in robotics is planning and selecting actions for goal-directed behavior in noisy uncertain environments. The problem is typically addressed within the framework of partially observable Markov decision processes (POMDPs). Although efficient algorithms exist for learning policies for MDPs, these algorithms do not generalize easily to POMDPs. In this paper, we propose a framework for planning and action selection based on probabilistic inference in graphical models. Unlike previous approaches based on MAP inference, our approach utilizes the most probable explanation (MPE) of variables in a graphical model, allowing tractable and efficient inference of actions. It generalizes easily to complex partially observable environments. Furthermore, it allows rewards and costs to be incorporated in a straightforward manner as part of the inference process. We investigate the application of our approach to the problem of robot navigation by testing it on a suite of well-known POMDP benchmarks. Our results demonstrate that the proposed method can beat or match the performance of recently proposed specialized POMDP solvers.**

## I. Introduction

Consider the problem of a delivery robot navigating in an office environment (Fig. 1) using a laser range finder. The robot may seek to navigate to specific locations such as the mail room from arbitrary locations in the office environment. The readings from the laser range finder are typically noisy and ambiguous, and therefore provide only partial information about the current location of the robot. Different locations in the environment may produce similar readings, resulting in what has been termed “perceptual aliasing.” How should the robot choose actions to maximize its probability of reaching a goal location, given only its noisy and ambiguous laser range finder readings?

Problems such as the robot navigation problem described above can be formalized within the framework of partially observable Markov decision processes (POMDPs) [1]. POMDPs are obtained from MDPs by incorporating observation probabilities that relate sensor readings (e.g., laser measurements) to hidden states (e.g., locations in the office). Although efficient algorithms exist for learning optimal policies (state to action mappings) for MDPs [2], [3], [4], [5], most of these algorithms do not generalize easily to POMDPs. Consequently, considerable effort has focused on using beliefs over states for approximate planning in POMDPs (e.g., [1], [6]).

In this paper, we investigate a new approach to planning and acting under uncertainty based on probabilistic inference in graphical models. The approach treats MDP and POMDP problems on an equal footing, and generalizes easily to other more complicated graphical models. We demonstrate the viability of our approach by solving several benchmark POMDP problems. The main contributions of our paper are: (1) an investigation of three different approaches to planning using probabilistic inference, (2) an exploration of three methods for probabilistic action selection and control, (3) a new algorithm for solving POMDP problems with applications to robotic navigation. Our approach opens the door to tackling a variety of problems in robotics ranging from feedback-based control to imitation using algorithms for exact and approximate inference in graphical models, in much the same way as viewing the deterministic planning problem as deduction [7] paved the way for the use of satisfiability solvers for efficient planning [8].

## II. Planning using Probabilistic Inference

### A. Notation

We use capital letters for variable names and small case letters to denote specific instances. Let $\Omega_S$ be the set of states in the environment and $\Omega_A$ the set of all possible actions available to the agent (both finite). Let there be a special single state $g \in \Omega_S$ called the “goal” state that the robot is trying to reach. Different episodes can have different goals. At time $t$, the robot is in state $S_t$ and executes action $A_t$, which changes the robot’s state in a stochastic manner given by the transition probability $P(A_{t+1} \mid S_t, A_t)$, which is assumed to be independent of $t$, i.e., $P(S_{t+1}=s' \mid S_t=s, A_t=a) = \tau_{s'sa}$. The parameters described above define a Markov Decision Process (MDP) [3]. We use capital letters (e.g., $S, A$) to denote the variables and lowercase letters (e.g., $s, a$) to denote specific instances. Also, when obvious from context, we use $s$ for $S_t=s$ and $a$ for $A_t=a$, etc.

### B. Classical Planning using MDPs

The problem of planning is classically stated as follows: Starting from an initial state $S_1 = s$ and a desired goal state $g$, compute a series of actions $A_{1:T}$ that maximizes the probability of reaching the goal state (note that there are no explicit rewards). Here, $T$ represents the maximum number of time steps allowed (the “episode length”). We do not require $T$ to be *exactly* equal to the shortest path to the goal, just an upper bound on the path length chosen based on a priori domain

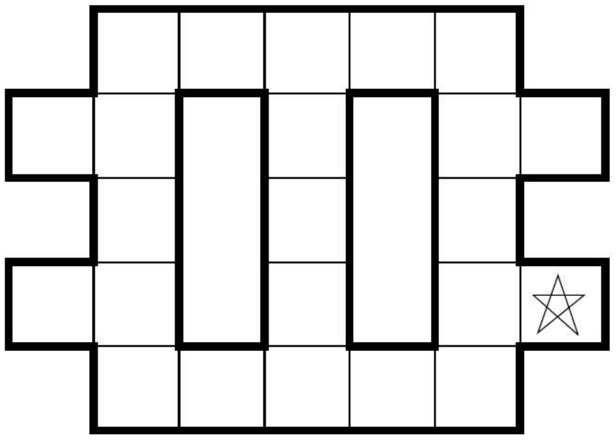

Fig. 1. **Navigating in an Uncertain Environment**. The figure shows an office environment used in the POMDP benchmark problem "Hallway2" [9]. The robot is required to navigate to the goal location marked with a star from arbitrary locations in the environment in the presence of considerable uncertainty due to ambiguity in laser range finder readings. Results obtained using our algorithm in this environment can be found in Section III-B

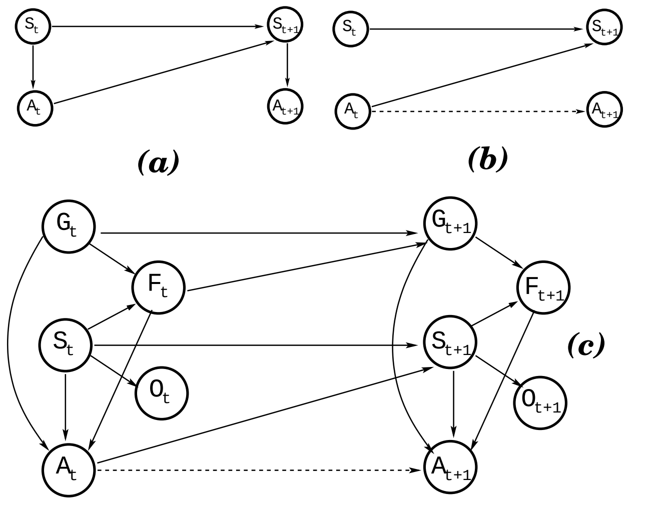

Fig. 2. **Graphical Models:** The graphical models (dynamic Bayesian networks or DBNs) used for planning and action inference. The figures show the network for two time slices. (a) The standard MDP (or FOMDP) model. (b) The Non Observable MDP (NOMDP) model: Dotted line represents a possible dependence between consecutive actions. (c) The POMDP model used in this paper. $O_t$ represents the observations recorded at time $t$. The goal node $G_t$ is set to the goal state at the beginning of an episode. The "finished" node $F_t$ is set to "0" unless the current state becomes equal to the goal state, when it is set to "1". When $F_t = 1$, the `stayput` action is preferred. This enables our approach to compute the shortest path to the goal, given a large enough value for $T$.

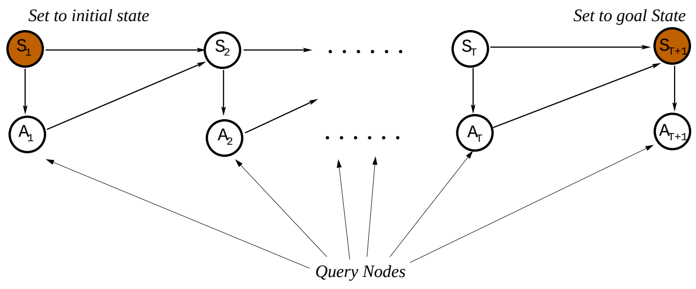

Fig. 3. **Planning as Probabilistic Inference of Actions in a Graphical Model**. The shaded nodes represent the evidence: initial state ($S_1$) and the desired goal state ($S_{T+1}$).

knowledge. The problem formulated above can be considered a special case of an MDP if we choose a suitable reward function. In particular, a reward structure can be imposed on the problem by assigning a high reward value to the goal state and zero or negative values for all other states. Solving the MDP involves learning a policy (state-to-action mapping) that maximizes total expected reward, a problem for which efficient algorithms exist [2], [3], [4], [5].

The policy learning problem can also be cast in terms of the graphical model in Fig. 2(a) where the task is to learn the optimal $S_t$ to $A_t$ conditional probability table. Although popular, policy learning suffers from the drawback that it needs to be re-learned each time there is a change in environment. More importantly, the MDP-based solutions are specific to the graphical model in Fig. 2(a) and do not generalize easily to more complex graphical models, especially those relevant to robotic applications such as navigating in real-world environments with noisy sensors. For example, even the simple alteration that results in the graphical model in Fig. 2(c), representing a POMDP model, causes problems.

### *C. Methods for Inferring Actions*

Instead of attempting to extend MDP-based methods to POMDPs, we tackle the problem of planning in partially observable uncertain environments using probabilistic inference in graphical models. The problem can be formulated as one of inferring an action $A_t$, given that $S_t = s$ and $S_{T+1} = g$ (the goal state) [10] (Fig. 3). The key difference between this approach and traditional MDP-based approaches is that it allows us to encode domain specific knowledge easily into the graphical model either as arcs encoding dependence between variables or as conditional probability tables (in particular priors over various states).

We consider three methods for action inference: (1) Computing the action maximizing the ***marginal distribution over actions*** at each time step, (2) Computing the ***maximum a posteriori (MAP) sequence of actions*** from the current state to the goal state, and (3) Computing the ***most probable explanation (MPE)*** of unknown variables, including action variables, in the graphical model unrolled in time.

*1) Marginal Distribution over Actions:* Algorithms for inference in graphical models typically compute the marginal distribution of unknown nodes in a graphical model given evidence on some nodes. This can be done efficiently for graphical models such as those in Fig. 2 using algorithms such as belief propagation. In our case, this approach reduces to computing the marginal distribution over the action nodes given the start and goal states:

$$\tilde{a}_t = \underset{a_t}{\operatorname{argmax}}\, P(A_t = a_t \mid S_t = s, S_{T+1} = g) \qquad (1)$$

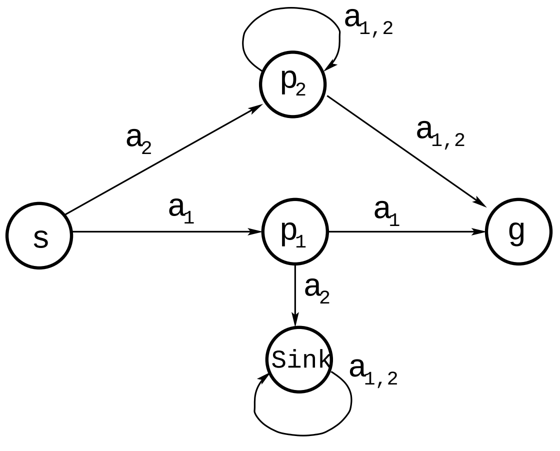

Fig. 4. **Sub-Optimality of the Marginal Method:** The figure provides a counterexample showing that the marginal action $\tilde{a}$ can be different from the optimal policy action $\hat{\pi}(s)$. The diagram is a compact representation of the transition matrix. All actions are deterministic except executing $a_1$ or $a_2$ in state $p_2$, both of which result in state $g$ with probability $\beta$ and back to $p_2$ otherwise.

One might be tempted to think that an optimal policy (state-to-action mapping) $\hat{\pi}$ can be easily learned by setting $\hat{\pi}(s) = \tilde{a}$. However, such a strategy may yield sub-optimal results. The marginal action is chosen so as to maximize the probability of success over *all* possible *futures*. It makes no assumption about potential optimal action(s) being executed after $\tilde{a_1}$ and hence sums over all possibilities in the future. An optimal policy, on the other hand, tries to find an action for $s$ which, when followed by other *optimal* actions, maximizes the expected probability of success.

To illustrate this point, consider the example in Fig. 4. There are five states: $s, g, p_1, p_2$ and $Sink$. $s$ is the start state $S_1$ and $g$ is the goal state. (want $S_{T+1} = g$). The transition probability table $\tau$ is defined as follows:

- For the start state $s$: $\tau_{p_1 s a_1} = 1$; and $\tau_{p_2 s a_2} = 1$ (Note: $\tau_{s' s a} = P(s'|s,a)$).
- Both $g$ and $Sink$ are "absorbing" states i.e. $\tau_{ssa} = 1$ for $s \in \{g, Sink\}$, for all $a$.
- For state $p_1$, $\tau_{g p_1 a_1} = 1$ and $\tau_{Sink p_1 a_2} = 1$.
- For state $p_2$, $\tau_{g p_2 a_{1,2}} = \beta$ and $\tau_{p_2 p_2 a_{1,2}} = 1 - \beta$.

Consider the optimal action for the start state $s$. The globally optimal policy action $\hat{\pi}(s) = a_1$. However, the marginal action is $a_2$ (as long as $\beta$ is not too low). To see why, consider the case where $T = 2$ i.e., $S_1 = s$, and $S_3 = g$. The marginal method computes the likelihood of the action $A_1 = a_1$ or $a_2$ according to $P(A_1|S_1 = s, S_3 = g)$. Assuming a uniform prior over actions, we get:

$$\begin{aligned}
& P(A_1|S_1 = s, S_3 = g) \propto P(A_1, S_1, S_3) \\
= & \sum_{A_2,S_2} P(S_1, A_1, S_2, A_2, S_3) \\
= & P(S_1)P(A_1|S_1)\sum_{A_2,S_2} P(S_2|S_1, A_1)P(A_2|S_2)P(S_3|S_2, A_2) \\
\propto & \sum_{S_2} P(S_2|S_1, A_1) \sum_{A_2} P(A_2|S_2)P(S_3|S_2, A_2)
\end{aligned}$$

Now, $P(A_2|S_2) = 0.5$ for all $S_2, A_2$ and $P(S_2|S_1, A_i) = \delta(S_2, p_i)$ for $i = 1, 2$. So the above expression simplifies to

$$P(A_1 = a_i|S_1 = s, S_3 = g) \quad \propto \quad \sum_{A_2} P(S_3 = g|p_i, A_2)$$

For $A_1 = a_1$ the above term reduces to 1 and for $A_2 = a_2$, the above term reduces to $2\beta$ since the goal is reached by both $A_2 = a_1$ and $A_2 = a_2$ with probability $\beta$. So if $\beta > 0.5$, the marginal action at $t = 1$ is $a_2$ where as the optimal policy $\hat{\pi}(s_1)$ is always $a_1$ as long as $\beta < 1$. The discrepancy arises because to compute the merit of $A_1 = a_1$, the marginal method sums over the possibility that $A_2$ might be $a_2$, which takes it to the $Sink$, where as the optimal policy is computed with the information that $\hat{\pi}(p_1) = a_1$.

*2) MAP Action Sequence:* A second approach to computing actions for reaching the goal state is to select the *maximum a posteriori* (MAP) action sequence:

$$\hat{a}_{1:T} = \underset{a_{1:T}}{\operatorname{argmax}} P(a_{1:T} \mid S_1 = s, S_{T+1} = g) \tag{2}$$

The MAP method was first suggested by Attias [10] who also proposed a polynomial time algorithm for this problem; however, the algorithm assumes a factorization (Equation (13) in [10]) which one may view as an approximation, leading to an approximate but not exact solution to the MAP problem. In fact, computing the exact MAP action sequence above can be shown to be an NP-complete problem (the proof is very similar to the proof presented in [11] showing that computing the MAP sequence in polytrees is NP-complete). Thus, it is unlikely that an efficient algorithm exists for computing the MAP sequence.

*3) MPE Action Sequence:* A tractable alternative to computing the MAP action sequence is to compute the most probable explanation (MPE) of variables in a graphical model, given a set of known variables:

$$\bar{a}_{1:T}, \bar{s}_{2:T} = \operatorname{argmax} P(a_{1:T}, s_{2:T} \mid S_1 = s, S_{T+1} = g) \tag{3}$$

The MPE method, which yields both action and state estimates, is inspired by the observation that humans often visualize a specific sequence of events when they plan, instead of optimally averaging over all outcomes as in the MAP method. Computing MPE is a straightforward inference problem that can be solved efficiently using standard techniques (such as the junction tree algorithm used for the results in this paper). It generalizes easily to *arbitrary* dynamic graphical models. The MPE method was first used in [12] to model human imitation. It is similar in spirit to the Most Likely State (MLS) heuristic used with some success in robot navigation, but differs from MLS in computing the most likely sequence of actions (in addition to states) to achieve a goal state.

## D. Action Selection and Control Strategies

If the state is observable at each time step, the robot has the option of using the current state information to re-plan at any time step. This leads to the following strategies for action selection and control:

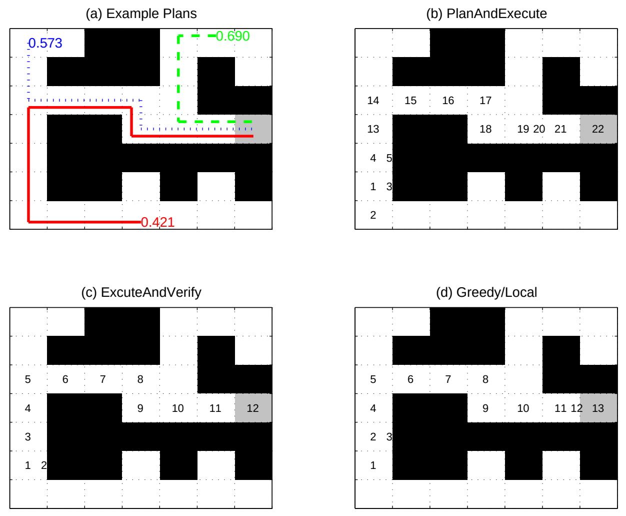

Fig. 5. **Planning and Action Selection:** (a) shows three example plans (action sequences) computed using the MPE method. The plans are shown as colored lines capturing the direction of actions. The numbers denote probability of success of each plan. The longer plans have lower probability of success as expected due to the noise in the environmental dynamics at each step. (b) Example of a path followed by the robot when utilizing the Plan-and-Execute strategy. Numbers denote states (locations) visited by the robot. (c) and (d) Representative paths of the robot when executing the Execute-and-Verify and Greedy/Local strategies respectively.

- **Plan-and-Execute ("Open Loop" Control):** Execute the the computed plan $a_{1:T}$. The plan could be the MAP estimate $\hat{a}_{1:t}$ or the MPE estimate $\bar{a}_{1:T}$. If the goal state is not reached, start with the current state as the new initial state and recalculate a new plan. Repeat until the goal state is reached. This constitutes an "open loop" control strategy where feedback from the environment is precluded during action execution. For the results discussed below, we used the MPE plan in the implementation of this strategy.
- **Greedy (Local Marginal Control):** The greedy strategy is to execute the local marginal action $\tilde{a}_t$ (Eq. 1) for the current time step and repeat until the goal state is reached.
- **Execute-and-Verify ("Closed Loop" Control):** Compute the MPE action and state sequence $\bar{a}_{1:T}, \bar{s}_{2:T}$ using Equation 3. At time step $t$, execute $\bar{a}_t$ and compare the new state $s^o_{t+1}$ with the predicted state $\bar{s}_{t+1}$. If the two states do not match, recompute the MPE sequences starting from $s^o_{t+1}$ and repeat. This strategy does not wait until the entire plan is executed before re-planning (as in Plan-and-Execute). It also does not necessarily require inference at each time step as in the Greedy method.

**Simulation Domain:** We illustrate the above strategies using a simulated office environment for the robot. The domain is similar to the standard stochastic "maze" domain [4], [10] (Figure 5). There are five possible actions: `forward`, `backward`, `left`, `right` and `stayput`. Each action takes the robot into the intended cell with a high probability and into the neighboring cells with probability $\eta$.

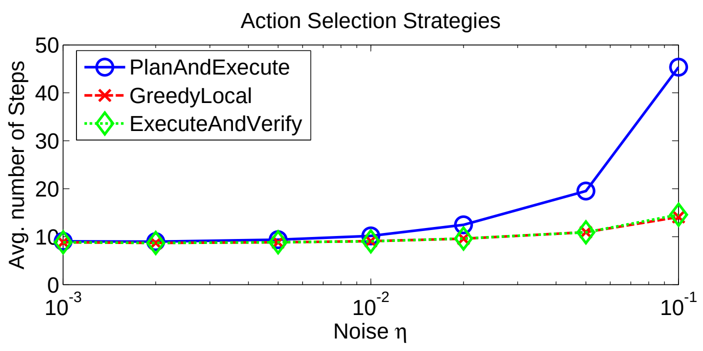

Fig. 6. **Comparison of Three Action Selection/Control Strategies:** The plots show average number of steps taken to reach the goal state as a function of noise $\eta$ for three different control strategies (see text). Each data point was computed by averaging over 500 runs.

Figure 5 shows some example MPE plans and the results of using the above control strategies for reaching a particular goal location (shaded). To quantify the efficiency of the three strategies, we calculated the average number of time steps taken to reach a fixed goal location from random initial locations. A uniform prior was used for $P(a_{1:T})$. As seen in Figure 6, while the Greedy and Execute-and-Verify strategies scale gracefully with increasing noise $\eta$, the Plan-and-Execute method takes exponentially more steps to reach the goal state because it requires that *all* actions succeed, the probability of which goes down exponentially ($< (1-\eta)^{PathLength}$) with increasing $\eta$. This is consistent with the intuition that Plan-and-Execute is a poor strategy for noisy environments where plans are prone to failure and where closed-loop strategies perform better. The Greedy and Plan-and-Execute methods give identical results in this experiment because for this simulated environment, both strategies reduce to finding the shortest path to the goal state, resulting in the same action for a given state. However, as discussed in Section II-C.1, there are cases where the Greedy method may not necessarily compute a globally optimal action.

## III. Partial Observability

The previous section assumes that the robot has access to the true state of the environment. For most practical domains, this assumption is not valid. The robot can only partially observe some aspect of the state (for example, location information through laser range finder readings) and the problem becomes a POMDP problem. If the observations give a very good estimate of the robot's state with little perceptual aliasing (e.g., GPS readings), one can solve underlying the MDP and solve the POMDP using the MLS (Most Likely State) heuristic [13]. However, in most realistic domains, the observations suffer heavily from perceptual aliasing, e.g., readings from a laser range finder in a robot for which similar rooms or hallways may produce near identical observations. In such cases, the strategy of learning the optimal policy for the underlying MDP is known not to work well. In this section, we propose a new algorithm for planning in POMDPs based on inferring actions through MPE.

## A. The POMDP algorithm

To take into account partial observability, we first add in the observation node $O_t$ to the graphical model (Fig. 2(c)). As before, we use the MPE estimate for $A_t$ as the next action to execute, given current and past observations and a desired goal state:

$$\bar{a}_{t:T}, \bar{o}_{t+1:T+1}, \bar{s}_{1:T} = \text{argmax} P(a_{t:T}, o_{t+1:T+1}, s_{1:T} \,|\, o_{1:t}, a_{1:t-1}, S_{T+1} = g) \quad (4)$$

Note that because the MPE method also provides estimates of expected observations[1], one only needs to compute the MPE estimate if the observation at $t+1$ is different than the one that was expected (cf. [14]). Thus, the method does not require re-planning at each time step (except in the worst case). Note also that since the algorithm is finding the MPE over the graphical model and is not dependent on any specific node, we can add a variety of constraints and priors on the graphical model (e.g., make it Semi-Markov) without altering the algorithm.

**Algorithm 1** MPE-based POMDP Algorithm using Graphical Models

```
1: Given: Initial observation $o_1$ , desired goal state $g$ and
   episode length $T$.
2: Compute $\bar{a}_{1:T}, \bar{o}_{1:T+1}$ as in Eq.4 for $t = 1$
3: for $t = 1$ to $T$ do
4:    Execute $\bar{a}_t$ to generate $O_{t+1} = o_{t+1}$.
5:    if $o_{t+1} \neq \bar{o}_{t+1}$ then
6:       Update $\bar{a}_{1:T}, \bar{o}_{1:T+1}$ as in Eq.4 for $t + 1$.
7:    end if
8: end for
```

The algorithm above computes actions based on all current and past observations and actions, and can handle non-trivial perceptual aliasing, unlike a previous inference algorithm [12] which (effectively) solved the POMDP by solving the underlying MDP.

## B. Results on Benchmarks

To evaluate the MPE-based POMDP algorithm described above, we ran it on two well-known benchmark navigation problems, Hallway and Hallway2, introduced in [1]. The objective is to reach a pre-specified goal state from random locations in the simulated environment. Information about location is obtained from simulated laser range finder measurements. The state spaces (locations) in these problems are reasonably large (57 and 89 respectively). We ran the experiments as 251 runs with $T$ set to 251 as in [1]. The results are shown in Figure 7: the numbers reflect the percentage of times the given method reached the goal location. As seen from the table, the MPE-based algorithm either beats or achieves the same degree of accuracy (100%) as state-of-the-art specialized POMDP solvers such as HSVI [15] and PBVI [16].

[1]MPE also estimates the most probable values for $G_t$ and $F_t$; these are omitted in Equation 4 for brevity.

| Domain | Q-MDP | PBVI | HSVI | MPE |
|---|---|---|---|---|
| Hallway | 47.4 | 96 | 100 | 100 |
| Hallway2 | 25.9 | 98 | 100 | 100 |

Fig. 7. **Comparison with other POMDP Algorithms on Benchmarks:** The numbers denote the percentage of times goal was reached starting from a location chosen randomly.

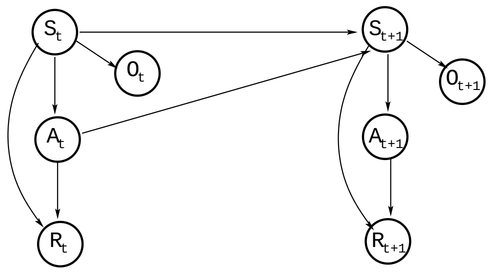

Fig. 8. **Incorporating Rewards in Inference-Based Planning:** The conditional probability table for boolean node $R_t$ incorporates rewards in the likelihood for the graphical model. See text for details. Nodes $G_t$ and $F_t$ have been dropped for the sake of simplicity.

## C. Incorporating Rewards and Costs

In many robotics applications, one needs to take into account the desirability or undesirability of intermediate states en-route to the goal. Additionally, some actions may be more costly to execute than others. Thus, the planning problem becomes one of selecting actions that maximize the probability of reaching a goal state while at the same time maximizing the total reward ((or minimizing the total cost) along the way. The challenge is to incorporate rewards and costs into graphical models in such a way that probabilistic inference automatically takes them into account, implementing a trade-off between reaching the goal state and maximizing total reward received within an episode. We solve this problem by introducing a new term encoding rewards in the likelihood for the graphical model [17].

Fig. 8 shows the new graphical model. The node $R_t$ is a binary node which is used to incorporate rewards into the likelihood. Let $\mathcal{R}(s, a)$ be the reward obtained on executing action $a$ in state $s$. Also, assume that $\mathcal{R}$ is normalized so that the maximum reward possible is 0, i.e $\mathcal{R}_{new}(s, a) = \mathcal{R}_{old}(s, a) - \mathcal{R}_{max}$ where $\mathcal{R}_{max} = \max_{s,a} \mathcal{R}_{old}(s, a)$.[2] The reward variable $R_t$'s conditional probability table (CPT) is defined to depend on $S_t$ and $A_t$ as follows:

$$P(R_t = 1|S_t = s, A_t = a) = e^{+\frac{1}{\lambda}\gamma^t \mathcal{R}(s,a)}$$

where $\lambda$ is a weighting factor and $\gamma \leq 1$ is an (optional) positive discount factor.[3] Since $\mathcal{R}(s, a) \leq 0$, the above value is between 0 and 1. Choosing the above CPT makes the likelihood of an action (and a state) proportional to the exponential of the reward received. During planning using MPE, we set $R_t = 1$ for all $t$ as "evidence" fed into the

[2]It is always possible to do this in the case of a *finite* horizon without changing the optimal solution.

[3]Usually, the CPT in a DBN is independent of $t$; however, most inference techniques allow different CPTs in different time slices.

graphical model. This causes maximizing the log likelihood to maximize the sum of rewards en-route to the goal state. More concretely, with $R_{1:T+1} = 1$[4], the log likelihood (LL) of the graphical model is given by:

$$\begin{aligned} & LL(S_{1:T+1}, A_{1:T+1}, R_{1:T+1} = 1) \\ = \quad & \frac{1}{\lambda}\left(\lambda LL(S_{1:T+1}, A_{1:T+1}) + \sum_{t=1}^{T+1} \gamma^t \mathcal{R}(S_t, A_t)\right) \end{aligned}$$

To gain an intuition about the above equation, assume that actions have deterministic affects. When $A_t$ is deterministic, the term $LL(S_{1:T+1}, A_{1:T+1})$ is $0$ for all invalid $S_{1:T+1}, A_{1:T+1}$ sequences and 1 for those which are consistent. This means that $LL(S_{1:T+1}, A_{1:T+1}, R_{1:T+1})$ reduces to $1+\frac{1}{\lambda}\sum_{t=1}^{T}\gamma^t\mathcal{R}(S_t, A_t)$. Finding the MPE estimate for $A_{1:T+1}$ then gives the optimal deterministic plan, i.e., the plan with the maximum reward. Note that the constant $C = \frac{1}{\lambda}$ represents the relative weight of maximizing the rewards versus maximizing the probability of reaching the goal. When $A_t$ has uncertain effects, the new LL is the weighted sum of the original LL and rewards obtained on the way to goal (note that this is different from the expected reward which requires averaging over states, a much more difficult problem).

We tested the above method for incorporating rewards by using an algorithm similar to Algorithm 1 on a set of reward-based POMDP benchmarks. The results (in terms of average reward obtained) are shown in Fig. 9. Once again, the general MPE-based algorithm (now with reward nodes) matches the performance of current state-of-the-art algorithms which were specifically designed to solve such POMDP problems.

## IV. Conclusion

This paper introduces a new approach to planning and action selection in partially observable environments based on probabilistic inference in graphical models. Because the approach is based on MPE estimation rather than MAP estimation, inference of actions is tractable. The approach can be used for both MDPs and POMDPs, as well as more complex graphical models. Additionally, the framework allows rewards and costs associated with states and actions to be incorporated within the inference process. The performance of the proposed approach was demonstrated using the problem of navigating to goal locations in noisy POMDP environments. On benchmark POMDP problems, our method's performance either beat or matched the performance of advanced POMDP solvers.

Our current efforts are focused on several outstanding issues. One important issue, especially for real-time performance, is speed of inference. The simulations in this paper employed Matlab/Java code based on the Bayesian Network Toolbox (BNT), which is not optimized for large networks. We anticipate considerable speed-up using C-code, sparse MPE inference, and other specialized inference algorithms. A second issue is understanding how well the approach scales to MDP and POMDP problems with larger numbers of states and actions. We are investigating one potential approach to handling large state spaces, namely, using a hierarchical extension of the current model. Such an extension would allow planning at multiple levels of abstraction in the state and action spaces, which could be especially beneficial for planning in noisy partially observable environments. Finally, we are exploring algorithms for inference in graphical models with continuous state and action spaces for applications such as controlling a robotic arm and maintaining balance in a humanoid robot.

[4]Even though we set $R_t = 1$ everywhere, we still model it as a random variable. This provides us with the option of ignoring rewards when the application demands maximization of the original goal-only likelihood.

| **Domain** | **Q-MDP** | **PBVI** | **HSVI** | **MPE** |
|---|---|---|---|---|
| Hallway | 0.26 | 0.53 | 0.52 | 0.51 |
| Hallway2 | 0.11 | 0.34 | 0.35 | 0.34 |

Fig. 9. **Comparison with other POMDP Solvers on Reward-Based Benchmarks:** The numbers denote the average rewards received starting from a location chosen randomly based on the belief state.

## References

[1] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up." in *ICML*, 1995, pp. 362–370.

[2] J. Blythe, "An overview of planning under uncertainty," *AI Magazine*, vol. 20(2), pp. 37–54, 1999. [Online]. Available: citeseer.ist.psu.edu/blythe99overview.html

[3] C. Boutilier, T. Dean, and S. Hanks, "Decision-theoretic planning: Structural assumptions and computational leverage," *Journal of AI Research*, vol. 11, pp. 1–94, 1999.

[4] R. S. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.

[5] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientifi c, 1996.

[6] K. Murphy, "A Survey of POMDP Solution Techniques," Comp. Sci. Div., UC Berkeley, Tech. Rep., 2000.

[7] C. Green, "Application of theorem proving to problem solving," 1969, pp. 219–239.

[8] H. A. Kautz and B. Selman, "Planning as satisfi ability," in *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)*, 1992, pp. 359–363.

[9] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *Proceedings of the Twelfth International Conference on Machine Learning*, A. Prieditis and S. Russell, Eds. San Francisco, CA, USA: Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1995, pp. 362–370.

[10] H. Attias, "Planning by probabilistic inference," in *Proceedings of the 9th Int. Workshop on AI and Statistics*, 2003.

[11] J. D. Park and A. Darwiche, "Complexity Results and Approximation Strategies for MAP Explanations," vol. 21, pp. 101–133, 2004.

[12] D. Verma and R. P. N. Rao, "Goal-based imitation as probabilistic inference over graphical models," in *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. Platt, Eds. Cambridge, MA: MIT Press, 2006.

[13] S. Koenig and R. Simmons, "Unsupervised learning of probabilistic models for robot navigation," *ICRA*, 1996.

[14] I. Nourbakhsh and M. Genesereth, "Assumptive planning and execution: a simple, working robot architecture," *Autonomous Robots Journal*, vol. 3, no. 1, pp. 49–67, 1996.

[15] T. Smith and R. Simmons, "Heuristic Search Value Iteration for POMDPs," in *Proc. of UAI 2004*, Banff, Alberta, 2004.

[16] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for pomdps," in *International Joint Conference on Artificial Intelligence (IJCAI)*, August 2003, pp. 1025 – 1032.

[17] D. Verma and R. P. N. Rao, "Solving MDPs and POMDPs using probabilistic inference over graphical models," University of Washington, Seattle, WA, Tech. Rep. UW-CSE-05-12-01, Dec. 2005.