Graphical Models for Planning and Imitation in Uncertain Environments

Deepak Verma DEEPAK@CS.WASHINGTON.EDU

Rajesh Rao RAO@CS.WASHINGTON.EDU

Department of CSE,
University of Washington,
Seattle, WA 98195 USA.

Technical Report 2005-02-01, Department of CSE, UW, February, 2005

Abstract

We show that the problems of planning, policy learning, and goal-based imitation can be addressed within a unified framework based on probabilistic inference in graphical models. Planning is viewed as probabilistic inference of an action sequence in a Dynamic Bayesian Network (DBN), given an initial state and a desired goal state. We describe how planning can be used to bootstrap the learning of goal-dependent policies by utilizing feedback from the environment and a learned environment model. We demonstrate several different strategies for plan execution and policy learning in the context of the standard navigation problem for a stochastic maze environment. In contrast to conventional planning and policy learning methods, our approach does not require a reward or utility function, although constraints such as shortest path to goal can be incorporated within the graphical model. Our results demonstrate that the approach can be used to handle the challenging case of partially observable states (e.g., POMDPs). To illustrate the versatility of the approach, we show that the same graphical model that was used for planning and policy learning can also be used for inferring the goals of an observed teacher, for executing actions under uncertain goals, and for imitating the teacher even when the demonstration by the teacher was incomplete.

1. Introduction

Two fundamental problems in AI are planning and policy learning. Planning involves computing a sequence of actions that will take an agent from its current state to a desired goal state. While classical planning typically dealt with deterministic environments, planning in uncertain environments has received considerable attention in recent years [DW91, BG00]. A parallel line of research has focused on learning policies [Bly99, BDH99] which prescribe the optimal action to take in any given state so as to maximize total future expected reward. If a model of the environment is available, learning the optimal policy reduces to straightforward dynamic programming (DP); in the absence of a model, methods from reinforcement learning [SB98, BT96] such as TD-learning and Q-learning have proved useful.

In this paper, we show that the problems of planning and policy learning can solved in a unified manner using inference in probabilistic graphical models. We demonstrate the applicability of the framework to the problems of goal inference and imitation. The main contributions of this paper are: (1) An efficient method for planning under uncertainty based on the most probable explanation (MPE) of hidden variables in a graphical model (cf. [Att03]), (2) A planning-based method for learning optimal policies that does not require an artificial reward structure to be imposed on the problem, (3) A new policy learning method for the challenging case of partially observable MDPs (POMDPs), (4) Methods for inferring the goals of an observed teacher, for acting under uncertain goals, and for goal-based imitation (offline and online) even in cases where the teacher provides an incomplete demonstration.

2. Graphical Models for Planning and Policy Learning

Let Ω_S be the set of states in the environment, Ω_A the set of all possible actions available to the agent, and Ω_G the set of possible goals. We assume all three sets are finite. For the present paper, we assume that goals represent states that the agent wants to reach. So each goal g represents a target state $Goal_g \in \Omega_S$. At time t the agent is in state s_t and executes action a_t. g_t represents the current goal that the agent is trying to reach at time t. Executing the action a_t changes the agent's state in a stochastic manner given by the transition probability $P(s_{t+1} \mid s_t, a_t)$, which is assumed to be independent of t i.e., $P(s_{t+1} = s' \mid s_t = s, a_t = a) = \tau_{s'sa}$.

Starting from an initial state $s_1 = s$ and a desired goal state g, the agent's aim is to reach the goal state by a series of actions $a_{1:T}$, where T represents the maximum number of time steps allowed (the "episode length"). Note that we do not require T to be *exactly* equal to the shortest path to the goal, just as an upper bound on the shortest path length. We use a, s, g to represent a specific value for action, state, and goal respectively. Also, when obvious from the context, we use s for $s_t = s$, a for $a_t = a$ and g for $g_t = g$. The problem formulated above can be considered a special case of a *Markov Decision Process* (MDP) [BDH99] if we choose a suitable reward function. As in the case of MDPs, the strategy to choose the optimal set of actions is critically dependent on the *observability* of the state. There exist three cases of interest:

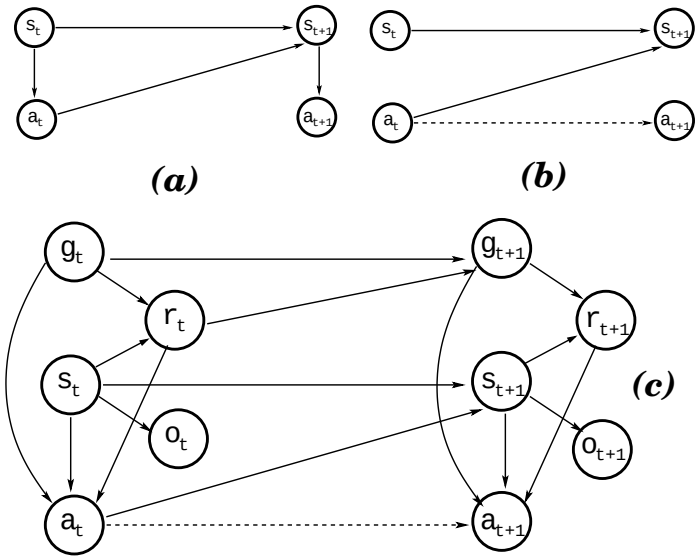

Figure 1: (a) The standard MDP (or FOMDP) model: The dependencies between the nodes from time step t to $t+1$ are represented by the transition probabilities and the dependency between actions and states is encoded by the policy. (b) The No Observations MDP (NOMDP) model: The agent receives no observations and must compute an entire action sequence through planning. Dotted line represents a possible dependence between consecutive actions. (c) The POMDP model (with goal and "reached" nodes) used in this paper.

Fully Observable MDP (FOMDP or just MDP): The state s_t is fully observed. The agent needs to compute a stochastic *policy* $\hat{\pi}_t(a \mid s, g)$ that maximizes the probability $P(s_{T+1} = Goal_g \mid s_t = s, g_t = g)$. For a large time horizon ($T \gg 1$), the policy is independent of t i.e. $\hat{\pi}_t(a \mid s, g) = \hat{\pi}(a \mid s, g)$ (a *stationary* policy). The graphical model for this case is shown in Fig. 1a. Goals are not explicitly shown in this graphical model because there is typically either only a single goal state or goal information is encoded in the reward model.
Non Observable MDP (NOMDP): In this case, the state s_t cannot be observed and the only information available to the agent is the initial state. The agent thus needs to compute an optimal *plan*, i.e.,a sequence of actions $\hat{a}_{1:T}$ that maximizes the probability $P(s_{T+1} = Goal_g \mid s_1 = s, a_{1:T})$. Assuming uniform prior over

actions this is same as maximizing $P(a_{1:T} \mid s_1 = s, g_t = g)$. Fig. 1b shows the graphical model for NOMDPs. We use the algorithm in [Att03] to compute the optimal plan for NOMDPs. This algorithm is also applicable if action a_t is conditioned on the previous action a_{t-1} as given by $P(a_t \mid a_{t-1})$ (dotted line in Fig. 1b). For the present paper, we set $P(a_t \mid a_{t-1})$ to uniform probabilities.

Partially Observable MDP (POMDP): In this case, the state s_t is hidden but it is possible to observe some aspects of it. Given the current state $s_t = s$, an observation o is produced with the probability $P(o_t = o \mid s_t = s) = \zeta_{so}$. In this paper, we assume the observations are discrete and drawn from the set Ω_O, although the approach can be easily generalized to the case of continuous observations (as in HMMs, for example). We extend the POMDP model to include a goal variable g_t and a "reached" variable r_t, resulting in the graphical model in Fig. 1c. The goal variable g_t represents the current goal the agent is trying to reach while the variable r_t is a boolean variable that assumes the value 1 whenever the current state equals the current goal state and 0 otherwise. In this paper, we use r_t to help infer the shortest path to the goal state (given an upper bound T on path length); this is done by constraining the actions that can be selected once the goal state is reached (see next section). Note that r_t can also be used to model the switching of goal states (once a goal is reached) and to implement hierarchical extensions of the present model. The current action a_t now depends not only on the current state but also on the current goal g_t, and whether we have reached the goal (as indicated by r_t). The FOMDP model is obtained as a special case of this model when $\Omega_O = \Omega_S$, $\zeta_{so} = \delta(s, o)$, and there is a single goal g.

The Maze Domain: To illustrate the proposed approach, we use the standard stochastic maze domain that has been traditionally used in the MDP and reinforcement learning literature [SB98, Att03]. Figure 2 shows the 7×7 maze used in the experiments. Solid squares denote a wall. There are five possible actions: `up`, `down`, `left`, `right` and `stayput`. Each action takes the agent into the intended cell with a high probability. This probability is governed by the noise parameter η, which is the probability that the agent will end up in one of the adjoining (non-wall) squares or remain in the same square. For example, for the maze in Fig. 2, $P([3, 5] \mid [4, 5], \texttt{left}) = \eta$ while $P([4, 4] \mid [4, 5], \texttt{left}) = 1 - 3\eta$ (we use `[i,j]` to denote the cell in `i`th row and `j`th column from the top left corner).

3. Planning and Plan Execution Strategies

In this section, we investigate solutions to the problem of planning an action sequence given the general graphical model in Fig. 1c. This would be used to boot strap policy learning. For simplicity, we assume full observability ($\zeta_{so} = \delta(s, o)$), although the solution generalizes easily to the partially observable case. We also assume that the environment model τ is known (the problem of learning τ is addressed in the next section). The problem of planning can then be stated as follows: Given a goal state g, an initial state s, and number of time steps T, what is the sequence of actions $\hat{a}_{1:T}$ that maximizes the probability of reaching the goal state? The *maximum a posteriori* (MAP) action sequence is:

$$\hat{a}_{1:T} = \operatorname*{argmax}_{a_{1:T}} P(a_{1:T} \mid s_1 = s, s_{T+1} = Goal_g) \tag{1}$$

Planning can thus be viewed as a probabilistic inference problem over a Dynamic Bayesian Network (DBN) [Att03]. The exact solution to inference of a subset of hidden variables in a Bayesian Network is known to be NP-complete [Coo90]. This problem is also a special case of solving a NOMDP problem with a specific reward function, an NP-complete problem in its most general form [PT87]. An efficient solution for a specific case was proposed recently by [Att03], but the approach does not generalize easily to arbitrary graphical models.

We propose a more tractable solution, namely, computing the most probable explanation (MPE) for a graphical model, given a set of known variables. This is a straightforward inference problem computable using standard techniques (such as the junction tree algorithm used for the results in this paper) and generalizes easily to arbitrary graphical models. When applied to the graphical model in Fig. 1c, our proposal for planning amounts to computing:

$$\bar{a}_{1:T}, \bar{s}_{2:T+1}, \bar{g}_{1:T}, \bar{r}_{1:T} = \text{argmax}\, P(a_{1:T}, s_{2:T}, g_{1:T}, r_{1:T} \mid s_1 = s, s_{T+1} = Goal_g) \tag{2}$$

Note that there may exist cases where $\bar{a}_{1:T} \neq \hat{a}_{1:T}$. However, the MPE-based plan is typically a good approximation to the MAP plan, is efficiently computable, and provides flexibility by allowing arbitrary graphical models. More importantly, the MPE-based plan can also be used to learn optimal policies, as described in section 4.

When using the MPE method, the "reached" variable r_t can be used to compute the shortest path to the goal. For $P(a \mid g, s, r)$, we set the prior for the `stayput` action to be very high when $r_t = 1$ and uniform otherwise. This breaks the isomorphism of the MPE action sequences with respect to the `stayput` action, i.e., for s_1 =`[4,6]`, goal=`[4,7]`, and $T = 2$, the probability of `right,stayput` becomes much higher than that of `stayput,right` (otherwise, they have the same posterior probability). Thus, the `stayput` action is discouraged unless the agent has reached the goal. This technique is quite general, in the sense that we can always augment Ω_A with a *no-op* action and use this technique based on r_t to push the *no-op* actions to the end of a T-length action sequence for a pre-chosen upper bound T.

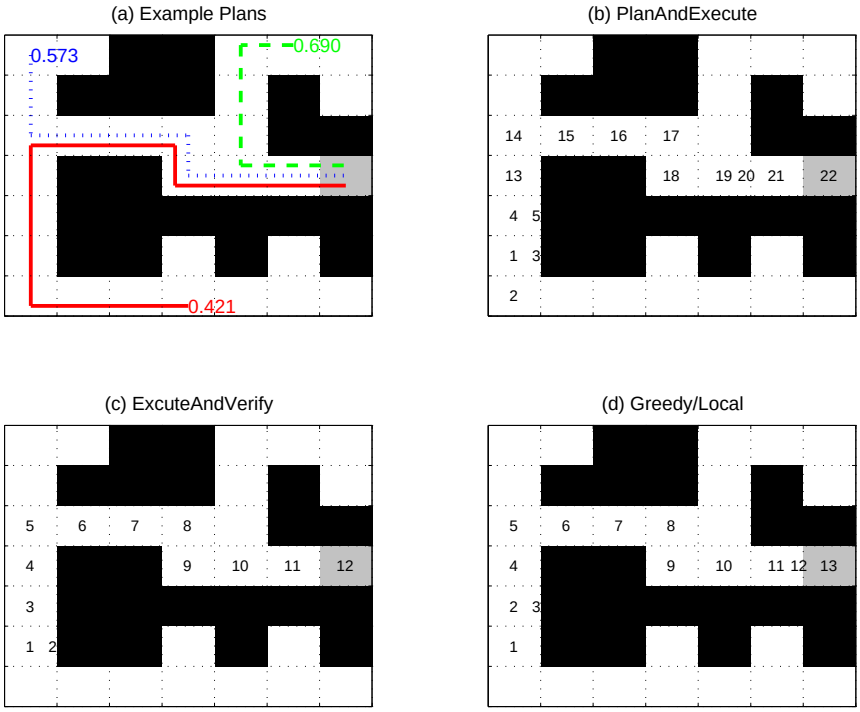

Figure 2: **Planning and Action Selection:** (a) shows three example plans (action sequences) computed using the MPE method. The plans are shown as colored lines capturing the direction of actions. The numbers denote probability of success of each plan. The longer plans have lower probability of success as expected. (b) Example of a path followed the agent when utilizing the Plan-and-Execute strategy. Numbers denote states (locations) visited by the agent. (c) and (d) Representative paths of the agent when executing the Execute-and-Verify and Greedy/Local strategies respectively.

The success of a plan is very sensitive to noise in the environment, as captured by the transition probabilities τ. A plan may fail when a single action in the plan "fails" (i.e., results in an unexpected state). This becomes more likely as plan length increases (see Fig. 2a). To examine this effect, we conducted the following experiment[1]: Starting from *each* of the valid locations in Fig. 2a, we computed the MPE plan $\bar{a}_{1:T}$ to

1. This experiment is similar to the one in [Att03] but since the maze structure and goal state were not described in that paper, our results cannot be compared to the results reported therein.

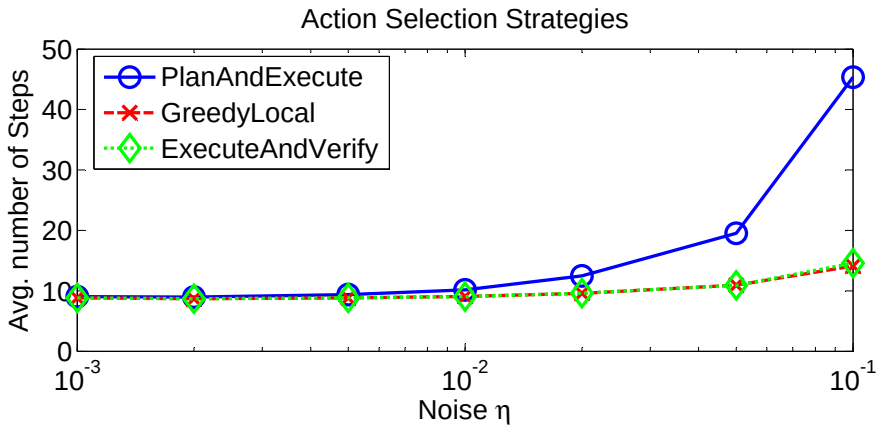

Figure 3: **Comparison of Three Action Selection/Control Strategies:** The plots show average number of steps taken to reach the goal state as a function of noise η for three different control strategies (see text). Each data point was computed by averaging over 500 runs.

reach the shaded goal location (`[4,7]`). For each such plan, we computed the the probability of reaching the goal state at or before time $T+1$ and averaged it over all initial locations. For noise parameter $\eta = 0.02$, the average probability of reaching the goal equals 0.619. It drops to 0.155 for $\eta = 0.1$. The corresponding numbers for the MAP plan $\hat{a}_{1:T}$ are 0.633 and 0.470, respectively. This suggests that the MPE method produces results comparable to the MAP method in the low noise regime while the MAP planning method is clearly better when the noise levels are high. However, when the noise levels are high, the probability of success becomes quite low and a policy-based method for reaching the goal state may be preferable (see section 4).

If the state is observable at each time step, an alternative to planning an entire action sequence *a priori* is to compute the MAP *action* at each time step based on the current state:

$$\tilde{a}_t = \operatorname*{argmax}_{a_t} P(a_t \mid s_t = s, s_{T+1} = Goal_g) \tag{3}$$

Eq. 3 can be efficiently computed by calculating the marginal probability for a_t, a standard computation in inference engines for graphical models.

Given the above algorithms for computing actions, we consider three action selection and control strategies for reaching a goal state:

- **Plan-and-Execute ("open loop" control):** Execute the MAP or MPE plan ($\hat{a}_{1:t}$ or $\bar{a}_{1:T}$). 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):** At each time step, execute $\tilde{a}_t$ as given by Eq. 3. Repeat until the goal state is reached.

- **Execute-and-Verify ("closed loop" control):** Compute the MPE plan and state sequence $\bar{a}_{1:T}, \bar{s}_{2:T}$ using equation 2. 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 replanning (as in Plan-and-Execute). It also does not necessarily require inference at each time step as in the Greedy method.

Results: We compared the three action selection/control strategies in terms of 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 3, while the Greedy and Execute-and-Verify strategies scale gracefully with increasing noise

η, 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 η. This is consistent with the intuition that Plan-and-Execute is a poor strategy for noisy domains 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 the maze problem, both strategies reduce to finding the shortest path to the goal state, resulting in the same action for a given state. However, there are cases where the Greedy method may not necessarily compute a globally optimal action (see the Appendix for an example).

4. Policy Learning

Executing a plan in a noisy environment may not always result in the goal state being reached. However, in the instances where a goal state is indeed reached, the executed action sequence can be used to bootstrap the learning of an optimal policy $\hat{\pi}(a \mid s, g)$, which represents the probability for action a in state s when the goal state to be reached is g. We define optimality in terms of reaching the goal using the shortest path. Note that the optimal policy may differ from the prior $P(a|s, g)$ which would count all actions executed in state s for goal g, regardless of whether the plan was successful.

For policy learning, the MPE planning method (Eq. 2) is preferable over the MAP method (Eq. 1) of [Att03]. This is because each individual action $\hat{a}_t$ in a MAP plan is chosen so as to maximize the probability of the entire sequence $a_{1:T}$ succeeding. In other words, $\hat{a}_t$ is the optimal action to take for a *specific* belief distribution obtained after executing $\hat{a}_{1:t-1}$ and to be followed by $\hat{a}_{t+1:T}$. As a result, $\hat{a}_t$ might not be the optimal local action given a particular state. For example, if s_1 =`[3,5]` and the goal is `[4,7]`, the MAP plan for $T = 5$ is `down,right,right,right,right`[2] while MPE plan is `down,right,right,stayput,stayput`. The MPE plan thus produces actions that are consistent with the optimal actions to take in a given state. In addition, we use the Plan-And-Execute method rather than the Greedy Method as the latter might yield actions that are not consistent with the globally optimal policy (see the Appendix for details).

MDP Policy Learning: Algorithm 1 shows a planning-based method for learning policies for an MDP (both τ and π are assumed unknown and initialized to a prior distribution, e.g., uniform). The agent selects a random start state and a goal state (according to $P(g_1)$), infers the MPE plan $\bar{a}_{1:T}$ using the current τ, executes it, and updates the frequency counts for $\tau_{s'sa}$ based on the observed s_t and s_{t+1} for each a_t. The policy $\hat{\pi}(a \mid s, g)$ is only updated (by updating the action frequencies) if the goal g was reached. To learn an accurate τ, the algorithm is biased towards exploration of the state space initially based on the parameter α (the "exploration probability"). α decreases by a decay factor γ ($0 < \gamma < 1$) with each iteration so that the algorithm transitions to an "exploitation" phase when transition model is well learned and favors the execution of the MPE plan.

POMDP Policy Learning: In the case of partial observability, Algorithm 1 is modified to compute the plan $\bar{a}_{1:T}$ based on observation $o_1 = o$ as evidence instead of $s_1 = s$ in Eq.2. The plan is executed to record observations $o_{2:T+1}$, which are then used to compute the MPE estimate for the hidden states: $\bar{s}^o_{1:T+1}, \bar{g}_{1:T}, \bar{r}_{1:T+1} = \text{argmax}\, P(s_{1:T+1}, g_{1:T}, r_{1:T+1} \mid o_{1:T+1}, \bar{a}_{1:T}, g_{T+1} = g)$. The MPE estimate $\bar{s}^o_{1:T+1}$ is then used instead of $s^o_{1:T+1}$ to update $\hat{\pi}$ and τ.

Experiments: We tested the above algorithm in the maze domain with $\eta = 0.02$ and three goal locations: `[4,7]`, `[2,7]` and `[1,2]`. The prior $P(g_1)$ was set to uniform probabilities, with α initialized to 1 and $\gamma = 0.98$. When computing the MPE plan during policy learning, the current policy $\hat{\pi}(a \mid s, g)$ was not substituted in the graphical model of Fig. 1c (a uniform $P(a|s, g)$ was assumed). This helped ensure that exploration was unbiased. For the POMDP case, we focused on learning $\hat{\pi}$ and assumed τ was given. We used $P(o_t \mid s_t) \propto e^{-3\|s_t - o_t\|}$ if o_t is valid where $\|s_t - o_t\|$ is the Manhattan distance between the observation o_t and state s_t on the maze. This yields about 0.85 probability of generating s_t as o_t and 0.15 distributed across nearby cells.

2. To see why, consider the situation in which $\hat{a}_2$ or $\hat{a}_3$ fails.

Algorithm 1 Policy learning in an unknown environment

Initialize transition model $\tau_{s'sa}$, policy $\hat{\pi}(a \mid s, g)$, α, and $numTrials$.
for $iter = 1$ to $numTrials$ **do**
  Choose random start location s_1 based on prior $P(s_1)$.
  Pick a goal g according to prior $P(g_1)$.
  With probability α:
    $a_{1:T}$= Random action sequence.
  Otherwise:
    Compute MPE plan as in Eq.2 using current $\tau_{s'sa}$.
    Set $a_{1:T} = \bar{a}_{1:T}$
  Execute $a_{1:T}$ and record observed states $s^o_{2:T+1}$.
  Update $\tau_{s'sa}$ based on $a_{1:T}$ and $s^o_{1:T+1}$.
  If the plan was successful, update policy $\hat{\pi}(a \mid s, g)$ using $a_{1:T}$ and $s^o_{1:T+1}$.
  $\alpha = \alpha \times \gamma$
end for

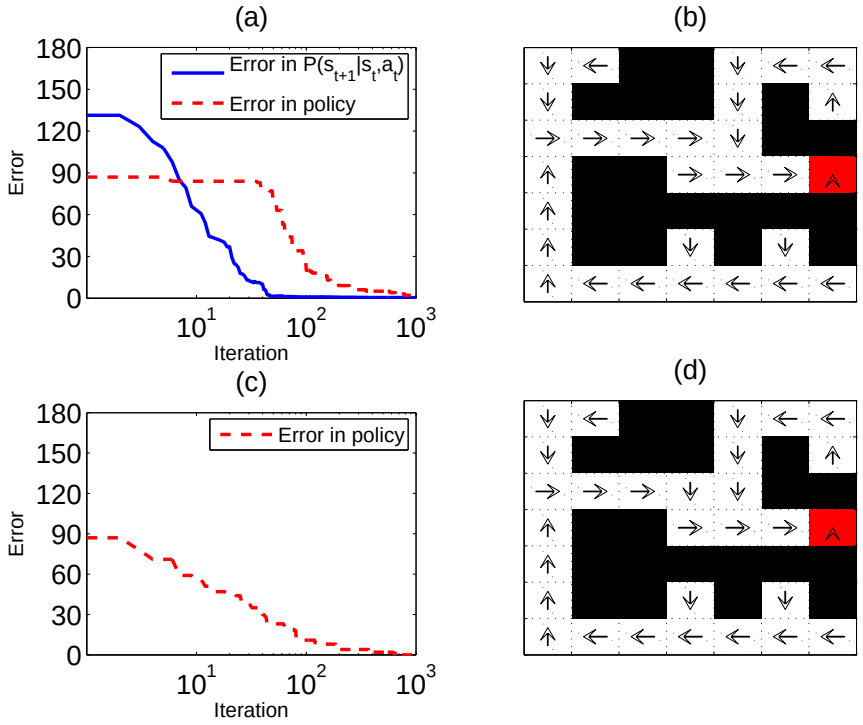

Figure 4: **Learning Policies for an MDP and a POMDP:** (a) shows the error in the transition model and policy w.r.t the true transition model and optimal policy for the maze MDP. (b) The optimal policy learned for one of the 3 goals. (c) and (d) show corresponding results for the POMDP case (the transition model was assumed to be known). The long arrows represent the maximum probability action while the short arrows show all the high probability actions when there is no clear winner.

Results: Figure 4a shows the error in the learned transition model and policy as a function of the number of iterations of the algorithm. Error in $\tau_{s'sa}$ was defined as the squared sum of differences between the learned and true transition parameters. Error in the learned policy was defined as the number of disagreements between the optimal deterministic policy for each goal computed via policy iteration and $\underset{a}{\operatorname{argmax}}\, \hat{\pi}(a \mid$

$s, g)$, summed over all goals. Both errors decrease to zero with increasing number of iterations. The policy error decreases only after the transition model error becomes significantly small because without an accurate estimate of τ, the MPE plan is typically incorrect and the agent rarely reaches the goal state, resulting in little or no learning of the policy. Figs. 4b shows the maximum probability action $\operatorname{argmax}_a \hat{\pi}(a \mid s, g)$ learned for each state (maze location) for one of the goals. It is clear that the optimal action has been learned by the algorithm for all locations to reach the given goal state. The results for the POMDP case are shown in Fig. 4c and d. Only policy error is plotted as the transition probabilities are given. Like the MDP case, the optimal policy is learnt and Fig. 4c shows the maximum probability action. As can be seen, the approach learns the right policy even in case of partial observability.

5. Inferring Goals and Goal-Based Imitation

Consider a task where the agent gets observations $o_{1:t}$ from observing a teacher and seeks to imitate the teacher. We model teacher observation as a partially observable MDP (POMDP) and use $P(o_t = o \mid s_t = s) = \zeta_{so}$ in Fig. 1c with ζ_{so} same as in Section 4. Also, for $P(a|s, g, r_t = 0)$, we use the policy $\hat{\pi}(a \mid s, g)$ learned as in the previous section.[3] The goal of the agent is to infer the intention of the teacher given a (possibly incomplete) demonstration and to reach the intended goal using its policy (which could be different from the teacher's optimal policy). Using the graphical model formulation the problem of goal inference reduces to finding the marginal $P(g_T \mid o_{1:t'})$. Imitation is accomplished by choosing the goal with the highest probability and executing actions to reach that goal.

Fig. 5a shows the results of goal inference for the set of noisy teacher observations in Fig. 5b. The three goal locations are indicated by red, blue, and green squares respectively. Note that the inferred goal probabilities correctly reflect the putative goal(s) of the teacher at each point in the teacher trajectory. In addition, even though the teacher demonstration is incomplete, the imitator can perform goal-based imitation by inferring the teacher's most likely goal as shown in Fig. 5c.

6. Online Imitation with Uncertain Goals

Now consider a task where the goal is to imitate a teacher online (i.e., simultaneously with the teacher). The teacher observations are assumed to be corrupted by noise and may include significant periods of occlusion where no data is available. The graphical model framework provides an elegant solution to the problem of planning and selecting actions when observations are missing and only a probability distribution over goals is available. The best current action can be picked using the marginal $P(a_t \mid o_{1:t})$, which can be computed efficiently for the graphical model in Fig. 1c. This marginal is equal to $\sum_i P(a_t|g_i, o_{1:t})P(g_i|o_{1:t})$, i.e., the policy for each goal *weighted* by the likelihood of that goal given past teacher observations, which corresponds to our intuition of how actions should be picked when goals are uncertain.

Fig. 6a shows the inferred distribution over goal states as the teacher follows a trajectory given by the noisy observations in Fig. 6b. Initially, all goals are nearly equally likely (with a slight bias for the nearest goal). Although the goal is uncertain and certain portions of the teacher trajectory are occluded[4], the agent is still able to make progress towards regions most likely to contain any probable goal states and is able to "catch-up" with the teacher when observations become available again (Fig.. 6c).

7. Conclusions

This paper proposes the use of graphical models for addressing a set of important problems including planning, policy learning, goal inference, and imitation within a single unified framework. We demonstrated the

3. A more versatile approach worthy of further investigation is to learn and use teacher-specifi c policies for imitation.
4. We simulated occlusion using a special observation symbol which carried no information about current state, i.e., $P(\text{occluded} \mid s) = \epsilon$ for all s ($\epsilon \ll 1$)

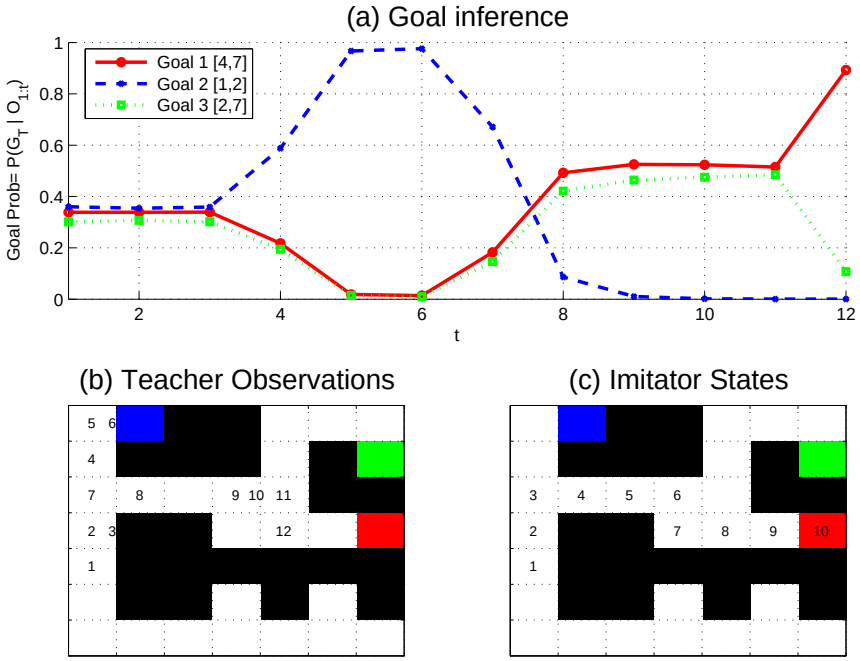

Figure 5: **Goal Inference and Goal-Based Imitation:** (a) shows the goal probabilities inferred at each time step from teacher observations. (b) shows the teacher observations, which are noisy and include a detour while en route to the red goal. The teacher demonstration is incomplete and stops short of the red goal. (c) The imitator infers the most likely goal using (a) and performs goal-based imitation while avoiding the detour (The numbers t in a cell in (b) and (c) represent o_t and s_t respectively).

viability of our approach in a simple stochastic maze navigation task. A major advantage of the proposed approach is its ability to handle partial information, especially the challenging case of POMDPs.

Our approach builds on the proposals of several previous researchers. It extends the approach of [Att03] from planning in a traditional state-action Markov model to a full-fledged graphical model involving states, actions, and goals with edges for capturing conditional distributions denoting policies. The indicator variable r_t used in our approach is similar to the ones used in some hierarchical graphical models [TMK04, FST98, BPV04]. However, these papers do not address the issue of action selection or planning. The proposed approach also extends a previous Bayesian model for imitation proposed in [RSM04], which assumed the availability of the complete teacher trajectory during demonstration and did not address the issue of planning.

Although our initial results are encouraging, several important questions remain. First, how well does the proposed approach scale to large-scale real-world problems? The relatively small state space of the maze navigation task greatly simplifies the problem of learning the environment model and the policies for various goal states. For larger state spaces, we intend to explore hierarchical extensions of the current model, potentially allowing planning at multiple time scales and policy learning at multiple levels of abstraction. A second line of research actively being pursued is the investigation of graphical models for continuous state and/or action spaces. Clearly, applications such as controlling a robotic arm or maintaining balance in a biped robot are better expressed in the continuous domain. We expect the insights gained from the current discrete state space model to be extremely helpful in formulating graphical models for planning and policy learning in continuous state spaces.

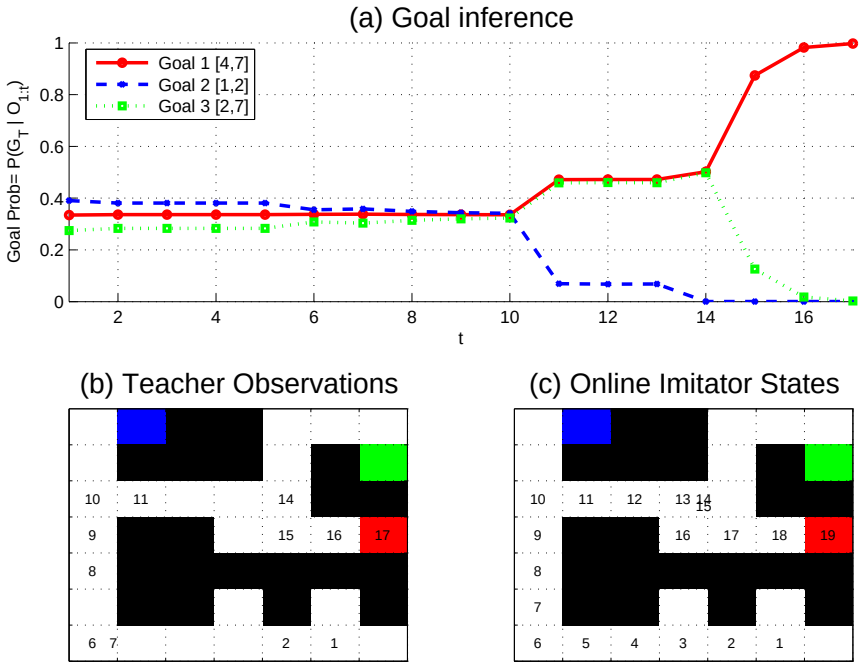

Figure 6: **Online Imitation with Uncertain Goals:** (a) shows the goal probabilities inferred by the agent at each time step for the noisy teacher trajectory in (b). (b) Observations of the teacher. Missing numbers indicate times at which the teacher was occluded. (c) The agent is able to follow the teacher trajectory even when the teacher is occluded based on the evolving goal distribution in (a).

References

[Att03] H. Attias. Planning by probabilistic inference. In *Proceedings of the 9th Int. Workshop on AI and Statistics*, 2003.

[BDH99] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of AI Research*, 11:1–94, 1999.

[BG00] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. In *Proc. 6th International Conf. on Artificial Intelligence Planning and Scheduling*, pages 52–61, 2000.

[Bly99] J. Blythe. An overview of planning under uncertainty. *AI Magazine*, 20(2):37–54, 1999.

[BPV04] H. Bui, D. Phung, and S. Venkatesh. Hierarchical hidden Markov models with general state hierarchy. In *AAAI 2004*, 2004.

[BT96] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[Coo90] G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artif. Intell.*, 42(2-3):393–405, 1990.

[DW91] T. L. Dean and M. P. Wellman. *Planning and control*. Morgan Kaufmann Publishers Inc., 1991.

[FST98] S. Fine, Y. Singer, and N. Tishby. The hierarchical hidden Markov model: Analysis and applications. *Mach. Learn.*, 32(1):41–62, 1998.

[PT87] C. Papadimitriou and J. N. Tsisiklis. The complexity of Markov decision processes. *Math. Oper. Res.*, 12(3):441–450, 1987.

[RSM04] R. P. N. Rao, A. P. Shon, and A. N. Meltzoff. A Bayesian model of imitation in infants and robots. In *Imitation and Social Learning in Robots, Humans, and Animals*. Cambridge University Press, 2004.

[SB98] R. S. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[TMK04] G. Theocharous, K. Murphy, and L. P. Kaelbling. Representing hierarchical pomdps as dbns for multi-scale robot localization. *ICRA*, 2004.

Appendix A. Sub-optimality of the Greedy Method

We presented three action selection strategies in the Section 3. We used the Plan-And-Execute strategy for learning the optimal policy. One might be tempted to think that an optimal policy can be easily learned by simply calculating the greedy action $\tilde{a}$ for each possible s (with $T \gg 1$) and setting $\hat{\pi}(s) = \tilde{a}$. However, such a strategy may yield sub-optimal results. The greedy 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. 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 the point, consider the scenario outlined in Fig.7:

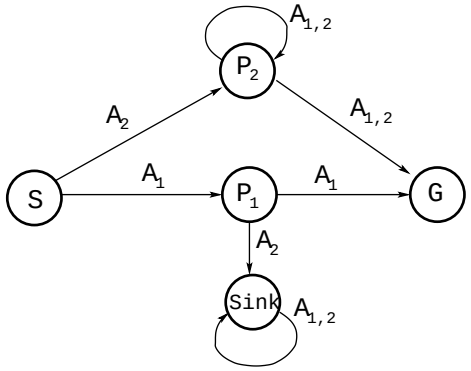

Figure 7: **Counterexample for showing that the Greedy Action $\tilde{a}$ can be different from the optimal policy action $\hat{\pi}(s)$.**

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. ($s_{T+1} = G$). The transition probability table τ is defined as follows:

- For the start state S: $\tau_{P_1 S A_1} = 1$; and $\tau_{P_2 S A_2} = 1$ (Remember that $\tau_{s' s s} = P(s'|s, a)$)
- 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 greedy action is $\tilde{a}_1 = A_2$ (as long as β is not too low). To see why, consider the case where $T = 2$ i.e., $s_1 = S, s_3 = G$. The greedy 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:

$$\begin{aligned} P(a_1|s_1 = S, s_3 = G) &\propto P(a_1, s_1, s_3) \\ &= \sum_{a_2, s_2} P(s_1, a_1, s_2, a_2, s_3) \end{aligned}$$

$$
\begin{aligned}
&= P(s_1)P(a_1|s_1)\sum_{a_2,s_2} P(s_2|s_1,a_1)P(a_2|s_2)P(s_3|s_2,a_2) \\
&\propto \sum_{s_2} P(s_2|s_1,a_1)\sum_{a_2} P(a_2|s_2)P(s_3|s_2,a_2)
\end{aligned}
$$

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 2β since the goal is reached by both $a_2 = A_1$ and $a_2 = A_2$ with probability β. So if $\beta > 0.5$, $\tilde{a}_1 = A_2$ else $\tilde{a}_1 = A_1$. Where as the optimal policy $\hat{\pi}(s_1)$ is always A_1 as long as $\beta < 1$. The discrepancy arises ,because, in the greedy method, to compute merit of $a_1 = A_1$, it has to sum over the possibility that a_2 might be A_2 which takes it to the $Sink$. Where as, while computing the optimal policy , the computation is done with the information that $\hat{\pi}(P_1) = A_1$ and hence prefers as A_1 for $\hat{\pi}(S) = A_1$ as there is some probability $(1 - \beta)$ of not reaching G when taking $a_1 = A_2$.