Planning by Probabilistic Inference

Hagai Attias
Microsoft Research
1 Microsoft Way
Redmond, WA 98052

Abstract

This paper presents and demonstrates a new approach to the problem of planning under uncertainty. Actions are treated as hidden variables, with their own prior distributions, in a probabilistic generative model involving actions and states. Planning is done by computing the posterior distribution over actions, conditioned on reaching the goal state within a specified number of steps. Under the new formulation, the toolbox of inference techniques be brought to bear on the planning problem. This paper focuses on problems with discrete actions and states, and discusses some extensions.

1 Introduction

Planning under uncertainty is a central problem in artificial intelligence and has applications in many domains. In the AI, control theory, and decision theory literature, planning (or sequential decision making) problems are often formulated using a dynamic Bayesian network with nodes corresponding to actions, states and rewards [1–3]. Such networks are termed influence diagrams or decision networks. Traditionally, the task has been defined as selecting an action sequence that maximizes the mean total reward. This requires searching the space of all possible action sequences and computing the mean total reward for each. Existing approaches to performing this search efficiently, starting with Bellman's work in the 1950s, seek an optimal policy, which is a mapping from states to actions. An optimal policy is computed using variants of the policy iteration or value iteration methods. Such methods work well for problems with fully observable states, where the states are discrete and the state space is relatively small. However, many cases of interest involve continuous or partially observable states, which pose difficult challenges for these methods. Whereas much recent work has attempted to meet these challenges (see, e.g., [4–7]), planning under uncertainty remains an open problem and the subject of active research.

This paper presents and demonstrates a new approach to selecting an optimal action sequence. It is motivated by the observation, which is perhaps a bit controversial, that the traditional approach may be attempting to solve a more difficult problem than necessary. First, many scenarios of interest do not inherently involve rewards. For instance, consider an agent navigating a maze, whose task is to reach an exit; or a chess playing agent, whose task is to defeat its opponent. The traditional approach adds to these problems a positive reward, given to the agent upon successfully completing the task, and perhaps additional positive and negative rewards along the way. The augmented problem may be formulated as a reward maximization problem. However, such tasks can also be formulated simply as reaching a state which is specified as a *goal state*; augmenting the problem with rewards is just a technical device which facilitates attacking them using policy/value iteration and their variants. Second, even in scenarios that inherently involve rewards, we suggest that the task may often be reformulated as reaching a goal state corresponding to maximal total reward.

In the following, we cast the problem of action selection as a problem of probabilistic inference in a generative model. We use models where actions are treated as hidden variables with their own prior distributions, and where one (or more) of the states are designated as goal states. We also specify a number N of actions the agent is allowed to take. We then compute the posterior distribution over the actions, conditioned on arriving at the goal state within N steps. The action sequence most likely to produce the goal state is the one which maximizes that posterior.

Here is the main contribution of our approach: under

the new formulation, the whole toolbox of inference techniques, both exact and approximate, can now be brought to bear on the planning problem, potentially resulting in many new algorithms for a variety of scenarios. In addition, our approach differs in important technical respects from existing methods. For example, whereas in the traditional framework, problems with partially observable states are intractable, in our approach some of these problems (see Section 7) are tractable, and are treated on the same footing as problems with fully observable states. For those problems that are intractable in our approach (see Section 8), a new class of algorithms may be derived based on approximate inference techniques.

This paper focuses on problems with discrete actions and states, and discusses some extensions.

2 The State-Action Model

Assume an agent may be in one of M states, and let s_n denote its state at time n. At each time, the agent may take one of K actions. Let a_n denote the action taken at time n. We assume that an action changes the agent's state in a stochastic manner. Taking action a_n in state s_n brings the agent to state s_{n+1} with probability $p(s_{n+1} \mid a_n, s_n)$. Starting from the initial state $s_1 = i$ and allowing the agent to act for N time points, the task of planning is to select an action sequence $a_{1:N} = (a_1, ..., a_N)$ that would take the agent to the goal state $s_{N+1} = g$.

Here we focus on the case where the states are completely observable, i.e., after taking action a_n, the state s_n is known with certainty. The case of partially observable states will be discussed later.

The planning problem can be cast in the form of an inference problem in a probabilistic generative model as follows. Define the transition probabilities

$$\begin{aligned} p(s_n = s' \mid s_{n-1} = s, a_{n-1} = a) &= \lambda_{s'sa} \ , \\ p(a_n = a' \mid a_{n-1} = a) &= \eta_{a'a} \end{aligned} \tag{1}$$

for $n = 2 : N + 1$, and

$$p(a_1 = a) = \eta_a \tag{2}$$

for $n = 1$ (note that we fix $s_1 = i$). Our model is defined by the joint probability over all states and actions, which is given by the product of the transition probabilities over time,

$$\begin{aligned} p(s_{2:N+1}, a_{1:N}) = & \prod_{n=2}^{N} p(s_n \mid s_{n-1}, a_{n-1}) p(a_n \mid a_{n-1}) \\ & \cdot p(a_1) p(s_{N+1} \mid s_N, a_N) \ . \end{aligned} \tag{3}$$

The model parameters are $\theta = \{\lambda_{s'sa}, \eta_{a'a}, \eta_a\}$.

Next, we consider the *posterior distribution over actions* by

$$p(a_{1:N} \mid s_1 = i, s_{N+1} = g) \ , \tag{4}$$

which is the distribution over actions conditioned on the initial state being i and the final state being the goal g. This distribution can be computed from our model via Bayes' rule. We define a *plan* as the action sequence $\hat{a}_{1:N}$ that maximizes the posterior over actions,

$$\hat{a}_{1:N} = \arg\max_{a_{1:N}} p(a_{1:N} \mid s_1 = i, s_{N+1} = g) \ . \tag{5}$$

Computing this sequence is referred to as planning.

Planning thus defined selects an action sequence which maximizes the probability of reaching the goal, with a bias toward a pre-defined action sequence. To see this, observe that the posterior over actions satisfies $p(a_{1:N} \mid s_1, s_{N+1}) \propto p(s_{N+1} \mid a_{1:N}, s_1) p(a_{1:N})$. Hence, the selected sequence maximizes a sum of two terms,

$$\begin{aligned} \hat{a}_{1:N} = \arg\max_{a_{1:N}} [& \log p(s_{N+1} = g \mid a_{1:N}, s_1 = i) \\ & + \log p(a_{1:N})] \ , \end{aligned}$$

where the first term is the log probability of arriving at the goal state g starting at the initial state i and taking N steps, and the second term is the log prior distribution over actions $p(a_{1:N})$. The prior over actions can be used not just to bias the plan toward a particular action sequence (the mean of the prior), but also to restrict the allowed actions to a particular regime, which may be achieved by setting the prior to zero (or making it vanishingly small) outside that regime.

One interesting point about our use of inference is that we are inferring hidden variables conditioned on data that we wish to observe in the future. Usually in generative models, data observed up to the present time is used to infer the value of hidden variables most likely to have generated the data. For instance, in classification problems, inference is used to identify the class label corresponding to the pattern at hand. But in our formulation of planning, inference identifies the action sequence that would be most likely to generate the desired goal state within N steps.

3 Planning Algorithm

We now provide an algorithm for computing an optimal plan, assuming the model parameters are known. Our model may be viewed as a hidden Markov model with all variables being hidden, except s_1 and s_{N+1}. The following algorithm can be shown to be equivalent to the Baum-Welch technique adapted to our case; the derivation is omitted.

We start by introducing the quantities $z_n(s,a)$ defined below. We also define the quantities $q_n(s',a' \mid s,a)$ and $q_n(a' \mid a)$ for $n = 2, ..., N$, and $q_1(a')$ for $n = 1$, by

$$q_n(s',a' \mid s,a) = \\ p(s_n = s', a_n = a' \mid s_{n-1} = s, a_{n-1} = a, s_{N+1} = g) \quad (6)$$

and

$$\begin{aligned} q_n(a' \mid a) &= p(a_n = a' \mid a_{n-1} = a, s_1 = i, s_N = g) \\ q_1(a') &= p(a_1 = a' \mid s_1 = i, s_{N+1} = g) \; . \end{aligned} \quad (7)$$

The z_n will turn out to be normalization constants of the q_n.

Backward pass. Initialize

$$z_{N+1}(s,a) = 1 \quad (8)$$

for all s, a. Then, for $n = N, ..., 2$ compute

$$\begin{aligned} q_n(s',a' \mid s,a) &= \frac{1}{z_n(s,a)} \lambda_{s'sa} \eta_{a'a} z_{n+1}(s',a') \\ z_n(s,a) &= \sum_{s'a'} \lambda_{s'sa} \eta_{a'a} z_{n+1}(s',a') \; . \end{aligned} \quad (9)$$

For $n = N$, the above equations are modified by a multiplicative factor of $\lambda_{gs'a'}$. Next, for $n = 1$ compute

$$\begin{aligned} q_1(a') &= \frac{1}{z_1} \eta_{a'} z_2(i,a') \\ z_1 &= \sum_{a'} \eta_{a'} z_2(i,a') \; . \end{aligned} \quad (10)$$

Forward Pass. For $n = 2, ..., N$ compute

$$q_n(s',a') = \sum_{sa} q_n(s',a' \mid s,a) q_{n-1}(s,a) \quad (11)$$

and

$$q_n(a' \mid a) = \frac{\sum_{s's} q_n(s',a' \mid s,a) q_{n-1}(s,a)}{\sum_s q_{n-1}(s,a)} \quad (12)$$

Planning. We now have

$$p(a_{1:N} \mid s_1 = i, s_{N+1} = g) = \prod_{n=1}^{N} q_n(a_n \mid a_{n-1}) \; . \quad (13)$$

The optimal sequence $\hat{a}_{1:N}$ (5) may be computed from it by the Viterbi algorithm.

4 Learning Algorithm

The next section discusses the mechanism for action selection. Here we assume that we have such a mechanism, and that the agent has used it to select a sequence (or several sequences) of actions. Using these actions and the resulting sequence of states, maximum likelihood estimation of the model parameters is straightforward. The agent learns $\lambda_{s'sa}$ and $\eta_{a'a}$ simply by counting,

$$\begin{aligned} \lambda_{s'sa} &= \frac{n_{s'sa}}{\sum_{s''} n_{s''sa}} \; , \\ \eta_{a'a} &= \frac{n_{a'a}}{\sum_{a''} n_{a''a}} \; , \end{aligned} \quad (14)$$

where $n_{s'sa}$ is the number of times taking action a at state s resulted in a transition to state s', and $n_{a'a}$ is the number of times the agent took action a' following action a.

It is also straightforward to incorporate prior knowledge (or bias) on these parameters, using prior distributions on the model parameters, $p(\lambda_{s'sa})$ and $p(\eta_{a'a})$. In our case where both states and actions are discrete, a convenient standard choice for each of these priors would be a Dirichlet distribution. A MAP estimate of the parameters using Dirichlet priors would modify the above estimates by an additional term. Posterior distributions over the parameters may also be computed, reflecting uncertainty in their estimates, as discussed in the last section.

5 Action Selection and Explore-Exploit

We introduce two primary modes of action selection. The first is *explore mode*, where actions are selected by sampling from a probability distribution. In this mode we describe two secondary modes. In *pure-explore mode* the agent samples from the prior over actions $p(a_{1:N})$. In *post-explore mode* the agent samples from the posterior over actions, conditioned on the initial and goal states (4). The second primary mode is *exploit mode*, where the selected action sequence maximizes that posterior as in (5).

These modes exhibit obvious differences when used to select actions after the learning phase has been completed. Actions selected in pure-explore mode do not reflect any information acquired by the model. In contrast, actions selected in post-explore and in exploit modes do reflect such information. Whereas exploit always selects the optimal action sequence, post-explore may select other actions as well.

More important are the differences between the modes when used during the learning phase. Notice that Eq. (5) would select an optimal action sequence after learning if, during learning, the agent has acquired enough information to predict with high probability that the sequence would bring it to the goal state. For this to happen, the agent should have tried that sequence, or nearby ones. If one uses pure-explore throughout

the learning phase, the model could learn the consequences of actions in a range as wide as the prior over actions allows. This would increase the accuracy of the posterior over actions (4) and facilitate selecting the optimal action sequence after learning. During learning, however, only occasionally would the agent select the optimal action sequence.

Compared to pure-explore, acting during learning in post-explore or exploit mode differs in two important respect. First, actions are selected from a narrower range, since the posterior is narrower than the prior. Second, this range changes with time, since the posterior is modified as additional data updates the model parameters. Hence, it depends on the initial conditions and on the initially selected actions. Now, if the agent happened to try actions near the optimal action sequence during learning, and discovered the optimal one, the posterior over actions would quickly start focusing on it. Thus, subsequent actions are likely to be optimal (or near optimal) as well, and so would be actions in exploit mode after learning. But, it may also happen that the agent never gets near the optimal sequence during learning. In that case, the agent will select suboptimal actions both during and after learning.

This discussion motivates one to use the following policy during learning. Start by acting in pure-explore, then switch to post-explore, using either a soft or a hard switch. One way to model this is by sampling actions from the distribution p_{learn}, defined by the mixture

$$\begin{aligned} p_{learn}(a_{1:N}) &= (1-\alpha)p(a_{1:N}) \\ &+ \alpha p(a_{1:N} \mid s_1 = i, s_{N+1} = g) \end{aligned} \quad (15)$$

where $0 \leq \alpha \leq 1$ is the switch. Note that one could also switch from pure-explore to exploit rather than to post-explore, which amounts to replacing the posterior over actions by a delta function centered at its maximum $\hat{a}_{1:N}$ (5). There are also ways other than (15) to model the switch, e.g., by sampling from p^1_{learn} defined by

$$\begin{aligned} p^1_{learn}(a_{1:N}) &= \frac{1}{Z_\alpha} p(a_{1:N})^{1-\alpha} \\ &\cdot \quad p(a_{1:N} \mid s_1 = i, s_{N+1} = g)^\alpha \end{aligned} \quad (16)$$

where $0 \leq \alpha \leq 1$ and Z_α is a normalization constant.

To summarize, an action selection mechanism is a function A. It takes as inputs an initial state i, a goal state g, a desired number of steps N, and a variable α interpolating between the pure-explore and post-explore modes. It returns an action sequence $a_{1:N}$,

$$a_{1:N} = A(i, g, N, \alpha) \; . \quad (17)$$

We point out one more twist on action selection during planning. The agent may call $A(i, g, N, \alpha)$ once and execute the full action sequence. Alternatively, it may take just a few actions, redraw the plan, take a few more actions, and repeat. To do that, it would call $A(i, g, N, \alpha)$, execute the first N' actions with $1 \leq N' < N$, observe the new state $s_{N'+1} = i'$, call again $A(i', g, N - N', \alpha)$, and repeat.

6 Results

We tested the algorithm on a probabilistic maze defined on a 7×7 grid, where each grid point represented a location. The agent could take 5 actions – north, east, south, west, and null. Each action brought the agent to the intended grid point with probability .7, and to any of the other 3 neighboring points with probability .1. The maze was constructed such that 20 grid points were forbidden, and so were the points surrounding the maze area. An attempted transition into a forbidden point resulted in a null transition. The maze had one exit.

In the first experiment, the transition parameters $\lambda_{s'sa}$ were fixed to their correct values, and the prior parameters were set to $\eta_{a'a} = .2$, corresponding to a uniform distribution. The agent was placed at an initial position and the task was to reach the exit within $N = 20$ steps. Averaged over 20 random initial positions, the success rate was .84. Using $N = 15$ decreased the success rate to .71.

In the second experiment, we used a random initialization for the transition parameters, and learned them by acting in pure-explore mode for 500 time points, followed by post-explore for additional 500 time points. Next, we repeated the previous experiment for $N = 20$. As expected, the success rate decreased, to .76. More experiments are necessary to investigate further the effects of exploration phases with different settings.

7 Partially Observable States

Until now, we have assumed that the states are fully observable, and the only non-deterministic feature of the model was which state s' resulted from taking action a in state s. However, in most cases of practical interest, only some aspects of the state are observable, and rest are hidden. For example, an agent equipped with sonar sensors that is navigating a space bounded by walls has no direct knowledge of its position (in x-y coordinates) in that space. It can only observe the sensor signals, which inform it how far ahead in several different directions it may find an obstacle. Furthermore, in general there is no one-to-one mapping between the observed and hidden features of a state.

In the case of the navigating agent, for instance, there are two reasons for this. First, depending on the configuration of the space, several different positions may yield the same sonar reading (e.g., corners of a symmetric room). Second, sonar signals are highly variable due to reflection properties of different objects, propagation properties of the medium, motion effects, and internal noise.

Existing planning techniques based on policy or value iteration become intractable when transitioning from fully to partially observable states. However, in our approach this transition is straightforward.

We now define the model with partially observed states. Whereas in our original model the state at time n was denoted by s_n and was fully observable, here we denote the state at time n by (s_n, x_n), where x_n is observed and s_n is now hidden. Henceforth, we will reserve the term *states* for the s_n, and use *data* or *observations* to refer to the x_n. As before, the agent may take one of K actions, and a_n denotes the action taken at time n. Here we assume that actions affect only the states, via the transition probability $p(s_n \mid a_{n-1}, s_{n-1})$, but this restriction may be easily relaxed. We also assume that the data at time n depend only on the state at that time. This dependence is probabilistic and described by the conditional distribution $p(x_n \mid s_n)$. Hence, our model components are

$$\begin{aligned} p(x_n = x \mid s_n = s) &= \mathcal{N}(x_n \mid \mu_s, \nu_s) \ , \\ p(s_n = s' \mid s_{n-1} = s, a_{n-1} = a) &= \lambda_{s'sa} \ , \\ p(a_n = a' \mid a_{n-1} = a) &= \eta_{a'a} \end{aligned} \tag{18}$$

with the obvious modification for $n = 1$. We use a Gaussian distribution with mean μ_s and precision (defined as the inverse covariance) matrix ν_s to describe the observations, which is defined by

$$\mathcal{N}(x_n \mid \mu_s, \nu_s) = \mid \frac{\nu_s}{2\pi} \mid^{1/2} e^{-(1/2)(x_n - \mu_s)^T \nu_s (x_n - \mu_s)} \ . \tag{19}$$

Other distributions may also be used. The full model is defined by the joint probability over all observations, states and actions, which is given by the product of the transition probabilities over time,

$$\begin{aligned} p(x_{1:N+1}, s_{1:N+1}, a_{1:N}) &= \prod_{n=1}^{N+1} p(x_n \mid s_n) \\ &\cdot \prod_{n=2}^{N+1} p(s_n \mid s_{n-1}, a_{n-1}) p(a_n \mid a_{n-1}) \\ &\cdot p(a_1) p(s_1) \ . \end{aligned} \tag{20}$$

The parameters of the new model are $\theta = \{\mu_s, \nu_s, \lambda_{s'sa}, \eta_{a'a}, \eta_a\}$.

The planning task is defined as follows. Starting from the initial observation x_1 and allowing the agent to act for N time points, the agent must select an action sequence $a_{1:N}$ that would take it to the goal state $s_{N+1} = g$. Note that we specify an initial *observation* but a final *state*. To see why, imagine that the agent is placed at a random position in a room, and is given a task to navigate to the exit door, which corresponds to the goal state g in our model. The agent observes the sonar signals x_1 but has no knowledge of its position s_1. Similarly, the observation x_{N+1} after N steps generally cannot determine whether the exit door has been reached, due to the variability of the sonar readings. This variability is captured by the distribution $p(x_{N+1} \mid s_{N+1} = g)$. It is therefore natural to define the task as reaching a particular state rather than a particular observation.

We point out that the hidden states do not need to correspond to physical variables in the problems. As an example, consider the case of the sonar using agent. Whereas the hidden states may represent the actual spatial locations of the agent, they may also represent clusters of locations, where different locations in a given cluster (e.g., different corners) produce similar sonar readings.

To perform planning, we consider the *posterior distribution over actions*, which is now conditioned on the initial observation and the final state,

$$p(a_{1:N} \mid x_1, s_{N+1} = g) \ . \tag{21}$$

This distribution can be computed from our model via Bayes' rule. Again, we define a *plan* as the action sequence $\hat{a}_{1:N}$ that maximizes the posterior over actions,

$$\hat{a}_{1:N} = \arg\max_{a_{1:N}} p(a_{1:N} \mid x_1, s_{N+1} = g) \ . \tag{22}$$

Planning and learning. The planning algorithm for this model is a direct extension of the planning algorithm presented above for the model with fully observable states, as inference remains tractable. The difference is that instead of the conditional distributions (6), one computes

$$\begin{aligned} &q_n(x', s', a' \mid s, a) = \\ &p(x_n = x', s_n = s', a_n = a' \mid \\ &\qquad s_{n-1} = s, a_{n-1} = a, s_{N+1} = g) \ , \end{aligned} \tag{23}$$

with similar simple modifications for the other q's.

The learning algorithm becomes a bit more complex compared to the case of fully observable states. Having taken actions a_n and observed the data x_n, the agent needs an EM algorithm to estimate the model parameters, as the states s_n remain hidden. This algorithm is straightforward to derive and is omitted.

7.1 Results

We tested this algorithm using a 5×5 gridworld bounded from all sides by a wall. The agent could take the same actions as in the previous experiments. The observation at each time point was a 4-dim vector, whose entries were the distance from the agent's location to the wall in the north, east, south, and west directions. The observed distance equaled the correct distance with probability .8, and differed from the correct distance by ± 1 grid point with probability .1. Starting at the southwest corner, the task was to reach the northeast corner. Only a preliminary experiment has been performed so far. Using a model with 4 hidden states, the model parameters was learned in pure-explore mode. Testing the learned model on the task, the success rate averaged over 10 trials was .81.

8 Incorporating Rewards

As has been pointed out in the introduction, many planning tasks discussed in the literature in terms of maximizing mean total reward can be formulated as requiring the agent to reach a particular goal state, in which case the approach presented so far may be applied. However, many other planning scenarios actually include a reward, which the agent is required to maximize. Here we outline a treatment of such scenarios within our approach.

We restrict the discussion to the case of fully observable states. As before, taking action a_{n-1} at state s_{n-1} brings the agent to state s_n with probability $p(s_n \mid s_{n-1}, a_{n-1})$. But in addition, the agent receives a reward r_n with probability $p(r_n \mid s_n, a_{n-1})$, which depends on the action taken and on the resulting state. The task is to choose an actions sequence $a_{1:N}$ that maximizes the total mean reward $\langle \sum_n r_n \rangle$, where $\langle \cdot \rangle$ averages w.r.t. the model distribution. This is the traditional formulation of the problem. We now proceed to discuss the new formulation.

First, instead of working with the reward r_n directly, we will be working with the *cummulative reward* $\bar{r}_n$ defined by

$$\bar{r}_n = \sum_{m=1}^{n} r_m \ . \tag{24}$$

Similarly, we use the probability of the cummulative reward, $p(\bar{r}_n \mid s_n, a_{n-1}, \bar{r}_{n-1})$. This probability (notice the dependence on $\bar{r}_{n-1}$) is derived from $p(r_n \mid s_n, a_{n-1})$ by observing that $\bar{r}_n = \bar{r}_{n-1} + r_n$. Hence, we have the model

$$p(\bar{r}_{1:N+1}, s_{1:N+1}, a_{1:N}) = \prod_{n=2}^{N+1} p(\bar{r}_n \mid s_n, a_{n-1}, \bar{r}_{n-1}) \cdot \prod_{n=2}^{N+1} p(s_n \mid s_{n-1}, a_{n-1}) p(a_n \mid a_{n-1}) \cdot p(a_1) p(s_1) p(\bar{r}_1 \mid s_1) \ . \tag{25}$$

Next, we propose a new task for the agent. Rather than maximizing the mean total reward $\langle \bar{r}_{N+1} \rangle$, consider the posterior over actions conditioned on the total reward $\bar{r}_{N+1}$, as well as on the initial state,

$$p(a_{1:N} \mid s_1 = i, \bar{r}_{N+1} = R) \ , \tag{26}$$

where R is a maximal value for the total reward to be discussed shortly. This distribution is obtained from our model via Bayes' rule. We define a *plan* as the action sequence $\hat{a}_{1:N}$ that maximizes the posterior over actions,

$$\hat{a}_{1:N} = \arg\max_{a_{1:N}} p(a_{1:N} \mid s_1 = i, \bar{r}_{N+1} = R) \ . \tag{27}$$

Planning thus defined selects an action sequence which maximizes the probability of achieving a total reward R, with a bias toward a pre-defined action sequence. To see this, observe that the posterior over actions satisfies $p(a_{1:N} \mid s_1, \bar{r}_{N+1}) \propto p(\bar{r}_{N+1} \mid a_{1:N}, s_1) p(a_{1:N})$. Hence, the selected sequence maximizes a sum of two terms,

$$\hat{a}_{1:N} = \arg\max_{a_{1:N}} [\log p(\bar{r}_{N+1} = R \mid a_{1:N}, s_1 = i) + \log p(a_{1:N})] \ .$$

It remains to select an appropriate value for R. In problems where the highest possible total reward may be estimated using available domain knowledge, this value may be substituted for R. Alternatively, the agent may use the following strategy. Plan to achieve a particular value of R. Record the actual total reward obtained. Next, plan to achieve a higher value of R. If the actual total reward obtained has increased, increase R and repeat. Iterate until the actual total reward stops increasing. Notice that, due to the probabilistic nature of the reward, care must be taken in defining the stopping criterion. We defer a discussion of this and related points to a separate paper.

Finally, we comment on implementation issues. Compared to the algorithms for planning to reach a goal state presented above, which work in the state-action space, an algorithm for achieving a desired total reward must work in the state-action-reward space. If the reward is discrete and assumes one of L values, the computational complexity increases by a factor of $\mathcal{O}(L)$. A large L would require using approximation techniques for inference in the model, in particular for computing the posterior (26). Such approximations may be based on clustering [10] or variational methods

[9]. Furthermore, if the reward is continuous, inference is computationally intractable even for a small L. For continuous rewards, our model belongs to the class of switching state space models, for which several approximation techniques have been developed [8,11,12].

9 Conclusion and Extensions

This paper has presented a new approach to planning under uncertainty. In our approach, actions are treated as hidden variables in a probabilistic generative model and have their own prior distributions. The model describes the joint distribution of actions and states. Goal planning is performed by computing the posterior distribution over actions, conditioned on initial and goal states. An extension to partially observable states and another extension to planning with rewards have been discussed.

This new framework opens a new avenue for research on planning /decision making under uncertainty. Here are several topics for further work. (1) Learning a 'policy'. Our state-action model used a prior distribution over actions where the action a_n depended only on the previous action a_{n-1}. However, one may add a dependence on the previous state s_{n-1} as well, and work with the conditional distribution $p(a_n \mid a_{n-1}, s_{n-1})$. As the agent selects actions in post-explore mode, this conditional distribution adapts to describe the dependence of an action on the state at which it is taken. To the extent that actions taken are optimal, this distribution would ultimately describe an optimal 'policy'. Consequently, optimal actions may be obtained by sampling from that distribution rather than by performing inference. (2) Extension to continuous states and actions. Except in the case of partially observable states whose observed part x_n was continuous, the states and actions in the models discussed in this paper were discrete. However, many scenarios of practical interest involve continuous actions (e.g., a robot's speed) and states (e.g., a robot's position). (3) Efficient inference methods. Having formulated the task of planning under uncertainty as an inference problem, a variety of exact and approximate inference methods may now be applied to this problem. In particular, new efficient approximate algorithms may be developed for planning in problems with a large state space, problems with continuous states and actions, and problems involving continuous rewards. (4) Bayesian treatment. Currently we use a MAP estimate for the model parameters. However, in cases where the amount of data collected by the agent is relatively small, there may be large uncertainty in our parameter estimates. A Bayesian treatment would compute the full posterior distribution over the parameters given the data, denoted here by $q(\theta)$. This distribution would then be used to compute the posterior over actions via

$$p(a_{1:N} \mid s_1 = i, s_{N+1} = g) = \int d\theta \; q(\theta) p(a_{1:N} \mid s_1 = i, s_{N+1} = g, \theta) \; , \tag{28}$$

thus incorporating parameter uncertainty into action selection. In many models, the computation of $q(\theta)$ and of the posterior over actions would be computationally intractable, making approximations such as those developed in [13,14] necessary. (5) Finding the shortest action sequence. Our current method finds the optimal N-action-long sequence, but not necessarily the shortest one. One approach may use a prior over actions which favors short sequences. (6) Investigating the relationship with MDP and POMDP based methods. In particular, the posterior over actions that we compute may be used to restrict the space of actions which traditional methods search to find optimal policies.

References

[1] C. Boutilier, T. Dean, S. Hanks (1999). Decision theoretic planning: structural assumptions and computational leverage. *JAIR* 1, 1-93.

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

[3] J. Pearl (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

[4] D. Koller, R. Parr (1999). Computing factored value functions for policies in structured MDPs. *Proc. IJCAI-99*.

[5] A.Y. Ng, M. Jordan (2000). Pegasus: A policy search method for large MDPs and POMDPs. *Proc. UAI-00*.

[6] D. Koller, R. Parr (2000). Policy iteration for factored MDPs. *Proc. UAI-00*.

[7] C. Guestrin, D. Koller, R. Parr (2001). Max-norm projections for factored MDPs. *Proc. IJCAI-01*, vol. 1, 673-680.

[8] Z. Ghahramani, G.E. Hinton (1998). Variational learning for switching state-space models. *Neural Computation* 12(4), 963-996.

[9] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, L.K. Saul (1999). An introduction to variational methods in graphical models. *Machine Learning* 37, 183-233.

[10] X. Boyen and D. Koller (1998). Tractable inference for complex stochastic processes. *Proc. UAI-98*, 33-42.

[11] D. Koller, U. Lerner, D. Angelov (1999). A Gen-

eral Algorithm for Approximate Inference and its Application to Hybrid Bayes Nets. *Proc. UAI-99*, 324-333.

[12] U. Lerner, R. Parr (2001). Inference in Hybrid Networks: Theoretical Limits and Practical Algorithms. *Proc. UAI-01*, 310-318.

[13] H. Attias (1999). A variational Bayesian methods for graphical models. *Proc. NIPS-99*.

[14] Z. Ghahramani, M.J. Beal (2001). Propagation algorithms for variational Bayesian learning. *Proc. NIPS-00*.