A General Model for Online Probabilistic Plan Recognition

Hung H. Bui
Department of Computing
Curtin University of Technology
PO Box U1987, Perth, WA 6001, Australia
URL: http://www.cs.curtin.edu.aurbuihh
Email: buihh@cs.curtin.edu.au

Abstract

We present a new general framework for online probabilistic plan recognition called the Abstract Hidden Markov Memory Model (AHMEM). The new model is an extension of the existing Abstract Hidden Markov Model to allow the policy to have internal memory which can be updated in a Markov fashion. We show that the AHMEM can represent a richer class of probabilistic plans, and at the same time derive an efficient algorithm for plan recognition in the AHMEM based on the Rao-Blackwellised Particle Filter approximate inference method.

1 Introduction

The ability to perform plan recognition can be very useful in a wide range of applications such as monitoring and surveillance, decision supports, and team work. However the plan recognizing agent's task is usually complicated by the uncertainty in the plan refinement process, in the outcomes of actions, and in the agent's observations of the plan. Dealing with these issues in plan recognition is a challenging task, especially when the recognition has to be done online so that the observer can react to the actor's plan in real-time.

The uncertainty problem has been addressed by the seminal work [Charniak and Goldman, 19931 which phrases the plan recognition problem as the inference problem in a Bayesian network representing the process of executing the actor's plan. More recent work has considered dynamic models for performing plan recognition online [Pynadath and Wellman, 1995; 2000; Goldmand *et* a/., 1999; Huber *et ai,* 1994; Albrecht *et* a/., 1998]. While this offers a coherent way of modelling and dealing with various sources of uncertainty in the plan execution model, the computational complexity and scalability of inference is the main issue, especially for dynamic models.

Inference in dynamic models such as the Dynamic Bayesian Networks (DBN) [Nicholson and Brady, 1994] is more difficult than in a static model. Inference in a static network utilizes the sparse structure of the graphical model to make it tractable. In the dynamic case, the DBN belief state that we need to maintain usually does not preserve the conditional independence properties of the single time-slice network, making exact inference intractable even when the DBN has a sparse structure. Thus, online plan recognition algorithms based on exact inference will run into problems when the belief state becomes too large, and will be unable to scale up to larger or more detailed plan hierarchies.

In our previous work, we have proposed a framework for online probabilistic plan recognition based on the Abstract Hidden Markov Models (AHMM) [Bui *et* a/., 2002]. The AHMM is a stochastic model for representing the execution of a hierarchy of contingent plans (termed *policies).* Scalability in *policy recognition* in the AHMM is achieved by using an approximate inference scheme known as the Rao-Blackwellised Particle Filter (RBPF) [Doucet *et al* 20001. It has been shown that this algorithm scales well w.r.t. the number of levels in the plan hierarchy.

Despite its computational attractiveness, the current AHMM is limited in its expressiveness, in particular, its inability to represent an uninterrupted sequence of plans and actions. This is due to the fact that each policy in the AHMM is purely reactive on the current state and has no memory. This type of *memoryless* policies cannot represent an uninterrupted sequence of sub-plans since they have no way of remembering the sub-plan in the sequence that is currently being executed. In other words, the decision to choose the next sub-plan can only be dependent on the current state, and not on the sub-plans that have been chosen in the past. Other models for plan recognition such that the Probabilistic State Dependent Grammar (PSDG) [Pynadath and Wellman, 2000; Pynadath, 1999] are more expressive and do not have this limitation. Unfortunately, the existing exact inference method for the PSDG in [Pynadath, 1999] has been found to be flawed and inadequate [Bui, 2002].

The main motivation in this paper is to extend the existing AHMM framework to allow for policies with memories to be considered. We propose an extension of the AHMM called the Abstract Hidden Markov mEmory Model (AHMEM). The expressiveness of the new model encompasses that of the PSDG [Pynadath and Wellman, 2000], thus the new model removes the current restriction of the AHMM. More importantly, we show that the RBPF approximate inference method used for the AHMM can be extended to the more general AHMEM as well, ensuring that the new generalized model remains computationally attractive. To the best of our knowledge, we are the first to provide a scalable inference method

for this general type of hierarchical probabilistic plan hierarchy.

The paper is structured as follows. Section 2 provides a more detailed discussion of the AHMM, PSDG and related models for online probabilistic plan recognition. The AHMEM is introduced in section 3 and the algorithms for plan recognition are presented in section 4. Experimental results with a prototype system are provided in section 5. Finally, we conclude and discuss directions for future work in section 6.

2 Related Models for Online Probabilistic Plan Recognition

In the AHMM [Bui *et al,* 2000; 2002], an agent's probabilistic plan is modeled by an abstract Markov policy (AMP). An AMP is an extension of a policy in Markov Decision Processes (MDP) defined within a subset of the environment state space so that it can select other more refined AMPs and so on to form a hierarchy of policies. The AMP is thus similar to a contingent plan that prescribes which sub-plan should be invoked at each applicable state of the world. The noisy observation about the environment state can be modelled by making the state "hidden", similar to the hidden state in the Hidden Markov Models [Rabiner, 1989]. The stochastic process resulting from the execution of an AMP is termed the *Abstract Hidden Markov Model.* Intuitively, the AHMM models how an AMP causes the adoption of other policies and actions at different levels of abstraction, which in turn generate a sequence of states and observations. In the plan recognition task, an observer is given an AHMM corresponding to the actor's plan hierarchy, and is asked to infer about the current policy being executed by the actor at all levels of the hierarchy, taking into account the sequence of observations currently available. This problem is termed *policy recognition* [Bui *et al,* 2002].

Scalability of policy recognition in the AHMM is achieved by using a hybrid inference method, a variant of the Rao-Blackwellised particle filter (RBPF) [Doucet *et al,* 2000]. When applied to DBN inference, Rao-Blackwellisation [Casella and Robert, 1996] splits the network into two sets of variables: the set of variables that need to be sampled (termed the Rao-Blackwellising (RB) variables), and the set of remaining variables whose belief state conditioned on the RB variables need to be maintained via exact inference (termed the Rao-Blackwellised (RB) belief state). The RBPF thus allows us to combine sampling-based approximate inference with exact inference to achieve efficiency and improve accuracy.

The Probabilistic State-Dependent Grammar (PSDG) [Pynadath, 1999; Pynadath and Wellman, 2000] can be described as the Probabilistic Context Free Grammar (PCFG) [Jelinek *et al,* 1992], augmented with a state space, and a state transition probability table for each terminal symbol of the PCFG. In addition, the probability of each production rule is made state dependent. As a result, the terminal symbol now acts like primitive actions and the non-terminal symbol chooses its expansion depending on the current state. The AHMM is equivalent to a special class of PSDG where only production rules of the form $X \rightarrow YX$ and A" $\rightarrow \emptyset$ are allowed. The first rule models the adoption of a lower level policy Y by a higher level policy A', while the second rule models the termination of the policy X. The PSDG model considered by Pynadath and Wellman allows for more general rules of the form $X \rightarrow Y_1 \ldots Y_m X$, i.e., the recursion symbol must be located at the end of the expansion. Thus in a PSDG, a policy might be expanded into a sequence of policies at the lower level which will be executed one after another before control is returned to the higher level policy.

Although more expressive than the AHMM, the existing computational method for inference with the PSDG remains inadequate. Pynadath proposed an exact method for updating the belief state of the PSDG in a "compact" closed form. The proposed algorithm seemingly gets around the exponential blow up in the size of the belief state. Unfortunately, the derivation of the algorithm is based on a flawed assumption that the higher levels in the belief state are independent of the lower levels given the current level. For more details about the flaw in the inference algorithm for PSDG, interested readers are referred to [Bui, 20021.

The AHMM, PSDG, and the proposed AHMEM are related to the Hierarchical Abstract Machines (HAM) [Parr, 1998] used in abstract probabilistic planning. In this model, the policy is represented by a stochastic finite automaton, which can call other automata at the lower level. Despite their representational similarity, the computational techniques for AHMEM and related models are intended for plan recognition whereas the HAM model is used for speeding up the process of finding an optimal policy for MDP.

If we ignore the state dependency, the DBN structure of the AHMEM and PSDG is similar to the structure of the Hierarchical Hidden Markov Model (HHMM) [Fine *et al,* 1998; Murphy and Pashkin, 2001]. However, while the HHMM is a type of Probabilistic Context Free Grammar (PCFG), the AHMEM and PSDG are not due to the state dependency in the model.

3 The Abstract Hidden Markov Memory Models

This section introduces the Abstract Hidden Markov Memory Models (AHMEM), an extension of the AHMM where the policy can have internal memory. Our main aim is to construct a general model for plan recognition whose expressiveness encompasses that of the current AHMM and PSDG models, while retaining the computational attractiveness of the AHMM framework. We first define the AHMEM in subsection 3.1. The DBN structure of the new model is given in subsection 3.2.

3.1 The Model

Consider an MDP-like model with S representing the states of the environment, and A representing the set of primitive actions available to the agent. Each action $a \in A$ is defined by its transition probability from the current state s to the next state s': $o_a(s,s')$. The set of abstract policies will include every primitive actions. Furthermore, the AHMM [Bui *et al,*

2000] defines higher level abstract policies on top of a set of policies as follows:

Definition 1 (Abstract Markov Policy (AMP)). Let Π be a set of AMPs, an AMP π^* over Π is defined as a tuple $(S_{\pi^*}, D_{\pi^*}, \beta_{\pi^*}, \sigma_{\pi^*})$ where:

- $S_{\pi^*} \subset \cup_{\pi \in \Pi} S_\pi$ is the set of applicable states.
- $D_{\pi^*} \subset \cup_{\pi \in \Pi} D_\pi$ is the set of destination states.
- $\beta_{\pi^*} : D_{\pi^*} \rightarrow (0, 1]$ is the set of stopping probabilities such that $\beta_{\pi^*}(d) = 1, \forall d \in D_{\pi^*} \setminus S_{\pi^*}$.
- $\sigma_{\pi^*} : S_{\pi^*} \times \Pi \rightarrow [0, 1]$ is the selection function where $\sigma_{\pi^*}(s, \pi)$ is the probability that π^* selects the policy π at the state s.

An AMP as defined above is purely reactive, in the sense that it selects the next policy at the lower level based only on the current state s. This restricts the set of behaviours that an AMP can represent. For example, it will not be able to represent a plan consisting of a few sub-plans, one followed by another regardless of the state sequence. To represent this kind of plans, the agent needs to have some form of internal memory to remember the current stage of execution. Let *M* be a set of possible internal memory states. We first extend the definition of our policy to include a memory variable which takes on values in *M* and is updated after each stage of execution of the policy.[1]

Definition 2 (Abstract Markov Policy with Memory (AMPE)). Let II be a set of AMPEs, an AMPE □* over II is defined as a tuple $(S_{\pi^*}, D_{\pi^*}, \beta_{\pi^*}, \sigma_{\pi^*}, i^e_{\pi^*}, \sigma^e_{\pi^*})$ where:

- $S_{\pi^*} \subset \cup_{\pi \in \Pi} S_\pi$ is the set of applicable states.
- $D_{\pi^*} \subset \cup_{\pi \in \Pi} D_\pi$ is the set of destination states.
- $\beta_{\pi^*} : D_{\pi^*} \times M \rightarrow$ (0,1] is the terminating probability. $\beta_{\pi^*}(d, m)$ is the probability that the policy π^* would stop if the current state is *d* and the current memory value is *m*.
- $\sigma_{\pi^*} : S_{\pi^*} \times M \times \Pi \rightarrow [0, 1]$ is the policy selection probability, a□ (.s, m,□) is the probability that π^* selects the policy □ at the state *s* and memory value *m*.
- $i^e_{\pi^*} : S_{\pi^*} \times M \rightarrow$ [0,1] is the initial distribution of memory values. $i^e_{\pi^*}(s, m)$ is the probability that the initial memory is m if □* commences at state *s*.
- $\sigma^e_{\pi^*} : S_{\pi^*} \times M \times M \rightarrow [0, 1]$ is the memory transition probability. $\sigma^e_{\pi^*}(s, m, m')$ is the probability that the next memory value is m' given that the current memory value is *m* and the current state is .s.

Subsequently, we will drop the subscript π^* if there is no confusion about the policy in the context. Note that all the states in $D \setminus S$ are called terminal states and thus $\beta(d, m) = 1, \forall m \in M, d \in D \setminus S$. Also, the policy selection, memory initial and transition probability have to be proper probability distributions, i.e. $\sum_{\pi \in \Pi} \sigma(s, m, \pi) = 1$, $\sum_{m \in M} i^e(s, m) = 1$,and $\sum_{m' \in M} \sigma^e(s, m, m') = 1$.

When an AMPE π^* is executed from a state .s, it first initialises its memory value m according to the distribution $i^e(s, m)$. Then a policy at the lower level π will be selected according to the distribution $\sigma(s, m, \pi)$. This policy π will be executed until it terminates at some state s'. At the new state .s', the policy π^* itself will terminate with probability $\beta(s', m)$. If it does not terminate, the memory variable will be given a new value m' according to the transition probability $\sigma^e(s', m, m')$. Then a new policy at the lower level π' is selected with probability $\sigma(s', m', \pi')$ and so on.

Using the AMPEs, we can construct a hierarchy of abstract policies in the same way as in the AHMM. We start with a set of primitive actions Π_0 = A, and for *k* = 1, . . . , *K* build a set of new AMPEs on top of Π_{k-1}. Then, if a top-level policy π^K is executed, it invokes a sequence of level-(K-l) policies, each of which invokes a sequence of level-(K-2) policies and so on. A level-1 policy will invoke a sequence of primitive actions which leads to a sequence of states. We can then introduce the hidden states and model the noisy observation of the state by an observation model $\Pr(o_t \mid s_t) = \omega(s_t, o_t)$. The dynamic process of executing a top-level AMPE is termed the Abstract Hidden Markov mEmory Model (AHMEM).

Some special cases of the AHMEM are worth mentioning here. First, the AHMM itself is a special AHMEM. The memoryless policies of the AHMM are equivalent to AMPEs where the dependency on the memory variable is ignored (e.g. when *M* is a singleton set). The class of PSDG considered in [Pynadath, 1999] can also be easily converted to an AHMEM. The terminal symbols in the PSDG are equivalent to the primitive actions. Each non-terminal symbol is equivalent to a memoryless policy. In addition, each sequence $Y = Y_1 Y_2 \ldots Y_n$ encountered on the RHS of a production $X \rightarrow Y_1 Y_2 \ldots Y_n$ or $X \rightarrow Y_1 Y_2 \ldots Y_n X$ is equivalent to a policy whose memory m taking on the values 1, . . . ,*n*. *Y* then simply selects the (memoryless) policy Y_i if $m = i$.

Note that in the AHMEM definition, we assume a balanced policy hierarchy for the ease of presentation (all the actions must appear at the same bottom level). However, we can also specify an unbalanced hierarchy by introducing some dummy policies which arc equivalent to primitive actions at the higher levels in the hierarchy.

3.2 DBN Representation of the AHMEM

The DBN structure of an AHM*E*M is very similar to that of an AHMM: at each time slice t, the variable s_t represents the current state, π^k_t, $k = 0, \ldots, K$ represents the current policy at all levels, e^k_t represents whether π^k_t terminates at the current time, and $l_t = \max\{k | e^k_t = T\}$ denotes the highest level of termination. In addition to these variables, we need to introduce the memory variable m^k_t for $k \geq 1$ to represent the current memory value of the policy π^k_t (the primitive action π^0_t has no memory). The 2-time-slice DBN is given in Fig. 1. The structure of this network is described below. Note that if x is a variable, we use the notation $x_{i:j}$ to denote the sequence $(x_i, x_{i+1}, \ldots, x_j)$, and similarly $x^{i:j}$ to denote $(x^i, x^{i+1} \ldots, x^j)$.

[1] One can argue that wc can always incorporate the memory variable in the environment state, and hence we gain no extra representational power just by introducing the memory variables. However, incorporating the memory variables in the state variable would blow up the size of the state space and thus defeat our purpose of keeping the model computationally feasible.

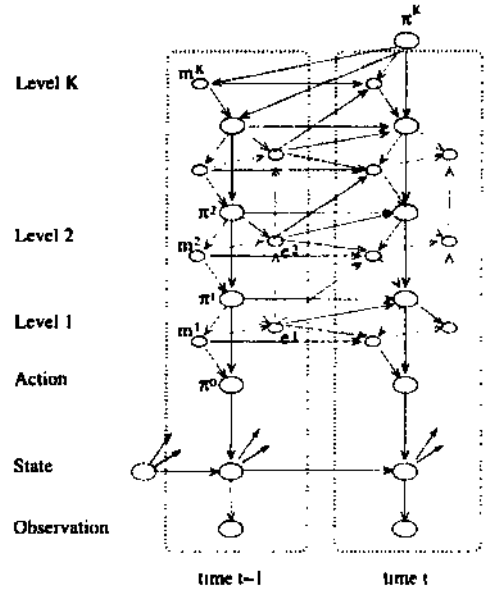

Figure 1: The 2-time-slice DBN for AHMEM

Policy Termination and Selection

The policy termination and selection model for the AHMEM is essentially the same as in the AHMM, except for the dependency on the value of the current memory variable m_t^k. We note that the same context specific independence [Boutilier *et al.*, 1996] properties of the AHMM still hold. For example, if $e_t^{k-1} = F$ then $e_t^k = F$ and e_t^k becomes independent of the remaining parents of this variable. Now consider the variable π_t^k. If $e_{t-1}^k = F$ then $\pi_t^k = \pi_{t-1}^k$ and π_t^k is independent of the remaining parents; otherwise, $\Pr(\pi_t^k \mid \pi_t^{k+1}, s_{t-1}, m_t^{k+1}) = \sigma_{\pi_t^{k+1}}(s_{t-1}, m_t^{k+1}, \pi_t^k)$ and π_t^k is independent of π_{t-1}^k.

Memory Update

Consider the variable m_t^k. There are two parents of this node that act like context variables: e_{t-1}^{k-1} and e_{t-1}^k. If $e_{t-1}^{k-1} = F$, the policy at the lower level has not terminated and memory is not updated at this time. Thus, $m_t^k = m_{t-1}^k$, and m_t^k becomes independent of all the remaining parents. If $e_{t-1}^{k-1} = T$ then there are two cases. If $e_{t-1}^k = F$, the policy at level k continues from the previous time, and thus the memory value will be updated according to the memory transition model: $\Pr(m_t^k \mid \pi_t^k, s_{t-1}, m_{t-1}^k) = \sigma_{\pi_t^k}^e(s_{t-1}, m_{t-1}^k, m_t^k)$. If $e_{t-1}^k = T$, the policy at level k has just started at state s_{t-1}, and thus the memory value will be initialised: $\Pr(m_t^k \mid \pi_t^k, s_{t-1}) = i_{\pi_t^k}^e(s_{t-1}, m_t^k)$. In this case, m_t^k is independent of the previous memory value m_{t-1}^k.

3.3 Independence Properties in the AHMEM

Even though AMPEs are more expressive than memoryless policies, they remain "autonomous", in the sense that the higher layers have no influence over the state of an AMPE during its execution. The only way the higher layers can influence the current state of an AMPE is through the conditions at the start: either through the starting state or the starting time. Thus, the conditional independence theorem for policies in the AHMM still holds in this more general setting. We state the theorem for the AHMEM below. The proof for the AHMM [Bui *et al.*, 2002] can be directly extended to this general case using the context specific independence properties described in the previous subsection.

Theorem 1. *Let τ_t^k and b_t^k be two random variables representing the starting time and the starting state, respectively, of the current level-k policy π_t^k: $\tau_t^k = \max\{t' < t \mid e_{t'}^k = T\}$ and $b_t^k = s_{\tau_t^k}$. Then given the policy π_t^k and its starting state and time, the set of current policies and memories at the higher levels are independent of the set of current policies, memories and state at the lower level:*

$$\pi_t^{k+1:K}, m_t^{k+1:K} \perp \pi_t^{0:k-1}, m_t^{1:k}, s_t \mid \pi_t^k, b_t^k, \tau_t^k \qquad (1)$$

Let $r_t = (s_t, l_t)$. Note that knowing $l_{1:t-1}$ is equivalent to knowing precisely when each of the policies starts and ends. Therefore, given $r_{1:t-1}$, the starting time and state of every current policy are known. The following corollary is thus a direct consequence of theorem 1.

Corollary 1. *Let C_t represent the conditional joint distribution $\Pr(\pi_t^{0:K}, m_t^{1:K}, s_t \mid r_{1:t-1})$. Then Ct has the following Bayesian network factorization:*

$$C_t = \left(\prod_{k=1}^{K} \Pr(\pi_t^k, m_t^k \mid \pi_t^{k-1}, r_{1:t-1})\right) \Pr(\pi_t^0, s_t \mid r_{1:t-1})$$

C_t thus also has the following undirected network representation. We first form the set of cliques: $c_0 = \{\pi_t^0, s_t\}$, $c_k = \{\pi_t^k, m_t^k, \pi_t^{k-1}\}$, $k = 1 \ldots, K$. Note that the set of cliques *CO-K* form a chain of cliques in this order, therefore we term *Ct* the *policy-clique chain.* This extends the concept of the *policy chain* in the memoryless case of the AHMM [Bui *et al,* 2000]. C_t can be factored into the product of potentials on these cliques: $C_t \propto \prod_{k=0}^{K} \psi_k(c_k)$. When $\psi_k = \Pr(\pi_t^k, m_t^k \mid \pi_t^{k-1}, r_{1:t-1})$ and $\psi_0 = \Pr(\pi_t^0, s_t \mid r_{1:t-1})$, i.e. when the potential factorization is the same as the directed network factorization, the potentials are said to be in *canonical* form. Any potential representation of the clique chain can be canonicalized by first perform message passing (exact inference) to compute the marginal at each clique. The canonical form can then be computed directly from these marginals. Later on we will use the undirected representation of *Ct* for exact inference, and the canonical form (directed representation) of *Ct* for obtaining samples from the joint distribution using simple forward sampling.

4 Approximate Inference for AHMEM

In this section, we look at the online inference problem in the AHMEM. Assume that at time f, we have a sequence of observations about the environment state $o_{1:t-1} = (o_1, \ldots, o_{t-1})$. We need to compute the belief state of the DBN which is the joint distribution of all the current variables given this observation sequence: $\Pr(\pi_t^{0:K}, m_t^{1:K}, s_t, l_t, o_t \mid o_{1:t-1})$. From this, we can answer various queries about the current status of the plan execution.

For example, the marginal probability of π_t^k tells us about the current policy the actor is executing at some level *k;* the probability P $(m_t^k \mid \pi_t^k)$l s us about the current stage of execution of a policy π_t^k; the probability $\Pr(e_t^k \mid \pi_t^k)$ tells us if π_t^k will end after the current time, etc.

Since there is no compact closed form representation for the above belief state, exact inference in the structure of the AHMEM is intractable when *K* is large. However, theorem 1 suggests that we can apply the Rao-Blackwellised Particle Filter (RBPF) to this problem in a similar way as in the AHMM [Bui *et al.,* 2002], i.e. by using r_t as the Rao-Blackwellising (RB) variables. The *Rao-Blackwellised (RB) belief state* is then similar to the original belief state of the AHMEM, except that now $r_{1:t-1}$ are known: $\mathcal{B}_t = \Pr(\pi_t^{0:K}, m_t^{1:K}, l_t, s_t, o_t \mid r_{1:t-1})$. Note that the RB belief state $\mathcal{B}_t$ can be obtained directly from the policy-clique chain *Ct* by adding in the network representing the conditional distribution of o_t and $e_t^{1:K}$, i.e. $\mathcal{B}_t$ — $\Pr(e_t^{1:K} \mid \pi_t^{0:K}, m_t^{1:K}, s_t)$ Pr(oi|st)Ct(seeFig.2).

Two main steps in the RBPF procedure are: (1) updating the RB belief state using exact inference, and (2) sampling the RB variable r_t from the current RB belief state.

4.1 Updating the RB Belief State

Fig. 2 shows the modified 2-time-slice DBN when the RB variables are known $(s_t = s, l_t = l)$. We note that all the nodes from above level / remain unchanged, while all the links across time slices from level / and below can be removed. This greatly simplifies the network structure, allowing the updating operations to be performed efficiently.

Since the B_t can be obtained directly from *Ct,* all we have to do is to compute $\mathcal{C}_{t+1}$ from C_t. The procedure for updating as usual has two stages: (1) absorbing the new evidence, i.e. from C_t, we need to compute $\mathcal{C}_{t+}$ — $\Pr(\pi_t^{0:K}, m_t^{1:K} \mid r_{1:t})$; and (2) projecting to the new time slice, i.e. from $\mathcal{C}_{t+}$, we need to compute *Ct+\.*

In principle, we apply a simplified version of the junction-tree algorithm Uensen, 1996] on the undirected network representation of *Ct* to perform the update step. This is in fact a generalization of the arc-reversal procedure which operates on directed network representation of the policy chain in the AHMM [Bui *et al,* 2002]. The algorithm for updating the RB belief state is given in Fig. 3. The absorbing step involves simply incorporating the evidence likelihood into the potentials of *Ct* to obtain the potentials for Ct+. The projecting step involves first adding time-slice *t* + 1 to Ct+ (see Fig. 2), then marginalizing the now redundant variables $\pi_t^{0:l}$, $m_t^{1:l+1}$ in the old time slice. The marginalization is done by performing message passing between the cliques of the 2-time-slice network shown in Fig. 2.

Since the potentials of C_t from level / + 2 to level K' stay unchanged, the complexity of the algorithm is *0(1)* where / is the highest level of termination. Furthermore, if one of these potentials is in canonical form, it remains in canonical form after the updating procedure.

4.2 RBPF for AHMEM

The full RBPF algorithm for policy recognition in the AHMEM is provided in Fig 4. The general structure is the same as the RBPF procedure for AHMM [Bui *et al,* 2002]. At each time step *t*, the algorithm maintains a set of *N* samples, each consists of a value for s_t~*i* and / / _ i , and a parametric representation of Ct The differences are in the details of how to obtain new samples and update the RB belief state.

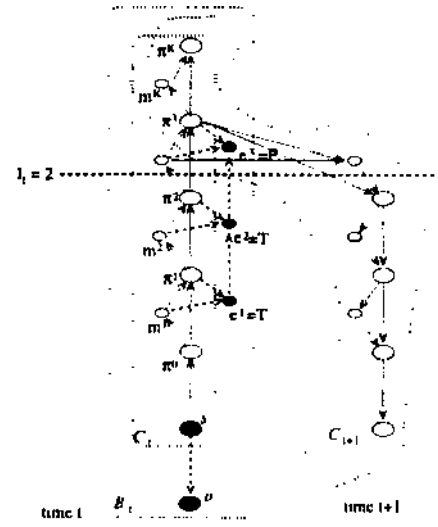

Figure 2: Two time slices of the RB belief state

To obtain each new sample *(s,,lt),* we first need to canonicalize the potentials of C_t. However, assuming that C_t-\ is in canonical form, we only have to canonicalize the potentials of C_t between level 0 and level *I* + 1 with complexity O(*1).* Thus the complexity of the sampling step is *O(Nl).* Since the complexity of the updating step is also *O(Nl),* the overall complexity of the algorithm at each time step is *O(Nl).* Furthermore, the distribution for *I* usually decays exponentially, thus the average complexity is only *O(N).*

If at some time *t,* an estimation e.g. $\Pr(\pi_t^K \mid o_{1:t-1})$ is required, we need to compute *h* = $\mathcal{C}_t(\pi_t^K)$ for each sample. This involves performing message passing for the entire chain *Ct.* Thus the complexity for this time step is *0(NK).* Other types of queries are also possible, as long as the probability required can be computed from the RB belief state. For example, we can ask the question: if the actor is currently executing π_t^k, what is the current stage of execution of this policy? To answer this query, we need to compute the conditional probability $\Pr(m_t^k \mid \pi_t^k, o_{1:t-1})$. This can easily be achieved by replacing the *ft* function in the algorithm with $h = \mathcal{C}_t(m_t^k \mid \pi_t^k)$.

5 Experimental Results

We have implemented the above algorithm in a surveillance domain to demonstrate its working in a practical application. The environment consists of a spatial area which has two separate rooms and a corridor monitored by a set of 5 cameras (Fig. 5). The monitored area is divided into a grid of cells, and the cell coordinates constitute the overall state space 5. The coordinates returned by the cameras are modelled as the noisy observations of the true coordinates of the tracked person, and over time, provide the sequence of observations *o\;t.*

Input: potentials for C_t, evidence s and l
Output: new potentials for C_{t+1}
Begin
/* absorbing */
$\psi_1 \leftarrow \psi_1 \times \psi_0(\pi_t^0, s)$
$\psi_{t+1} \leftarrow \psi_{t+1} \times (1 - \beta_{\pi_t^{l+1}}(s, m_t^{l+1}))$
For $k = 1, \ldots, l$
$\psi_k \leftarrow \psi_k \times \beta_{\pi_t^k}(s, m_t^k)$
/* projecting */
Construct the new cliques:
$c_{t+1}^e = \{m_t^{l+1}, m_{t+1}^{l+1}, \pi_t^{l+1}\}, c_{t+1}^l = \{\pi_t^{l+1}, m_{t+1}^{l+1}, \pi_{t+1}^l\}$
$\psi_{t+1}^e = \sigma_{\pi_t^{l+1}}^e(s, m_t^{l+1}, m_{t+1}^{l+1}), \psi_{t+1}^l = \sigma_{\pi_t^{l+1}}(s, m_{t+1}^{l+1}, \pi_{t+1}^l)$
Perform message passing along the path
$c_1 \rightarrow c_2 \rightarrow \ldots \rightarrow c_{t+1} \rightarrow c_{t+1}^e \rightarrow c_{t+1}^l$
$\psi_{t+1} = \psi_{t+1}^l$
$\psi_0 = \sigma_{\pi_{t+1}^0}^0(s, s_{t+1})$
For $k = 1 \ldots l$
$\psi_k = \sigma_{\pi_{t+1}^k}(s, m_{t+1}^k, \pi_{t+1}^{k-1}) \mathfrak{z}_{\pi_{t+1}^k}^e(s, m_{t+1}^k)$
End

Figure 3: Updating C_t

Begin
For $t = 1, 2, \ldots$
/* sampling step */
For each sample $i = 1, 2, \ldots, N$
Absorb o_t into $C_t^{(i)}$.
Canonicalize $C_t^{(i)}$, compute $w_t = \mathcal{B}_t(o_t)$.
Sample $s_t^{(i)}, l_t^{(i)}$ from $C_t^{(i)}$ and $\Pr(e_t^{1:K} \mid s_t, \pi_t^{0:K}, m_t^{1:K})$
Update weight $w^{(i)} = w_t$
/* re-sampling step */
Normalize the weight $\tilde{w}^{(i)} = \frac{w^{(i)}}{\sum_{i=1}^N w^{(i)}}$
Re-sample the sample set according to $\tilde{w}^{(i)}$
/* exact step */
For each sample $i = 1, 2, \ldots, N$
Compute $C_{t+1}^{(i)}$ from $C_t^{(i)}$ and $s_t^{(i)}, l_t^{(i)}$
Compute $h^{(i)} = C_{t+1}^{(i)}(\pi_{t+1}^k)$
/* Estimation step */
Compute the estimator $\Pr(\pi_{t+1}^k \mid o_{1:t}) \approx \hat{f} = \frac{1}{N} \sum_{i=1}^N h^{(i)}$
End

Figure 4: RBPF for AHMEM

At the top-level, each policy models a type of activity performed by the person. For example, *print* is a policy that involves a sequence of going to the computer and the printer, possibly going to the paper store if the printer is out of paper, and exiting. Note that this policy cannot be represented in the AHMM framework. In the AHMEM, this can be represented by a policy whose memory transition model is given in Fig. 6. The policies at the lower level, such as going to a computer, model the person's trajectories toward a set of special landmarks in the environment. These policies are constructed as memoryless policies in the same way as in the AHMM [Bui *etal,* 2002].

The result of querying the top-level policy for the trajectory in Fig. 5 is given in Fig. 7. The system correctly identifies the most probable policy as *print.* The result of querying the memory variable of this policy is given in Fig. 8. The system correctly identifies the sequence of lower-level policies invoked by the *print* policy. These results are obtained by running the RBPF algorithm with 3000 samples. The average processing time for each observation is approximately 0.9 seconds on a 1.7GHz desktop machine. For a more detailed description of this surveillance system, readers are referred to [Nguyen *et al.,* 2003].

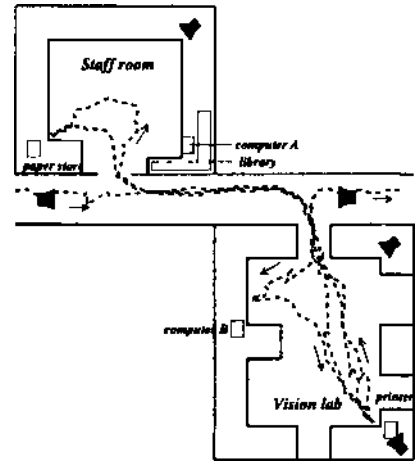

Figure 5: The environment and a person's trajectory

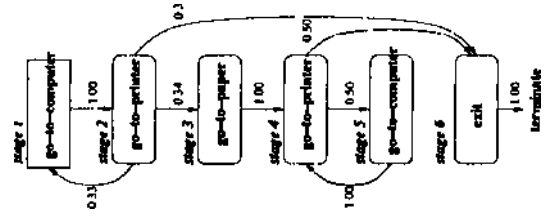

Figure 6: Memory transition of the policy "print"

6 Conclusion

In conclusion, we have presented the Abstract Hidden Markov Memory Models (AHMEM), a general framework for representing and recognizing probabilistic plans online. The new framework extends the AHMM by allowing the policies to have internal memories which are updated in a Markov fashion. This allows the AHMEM to represent a richer set of hierarchical probabilistic plans, including those belonging to the class of PSDG previously considered. Furthermore,

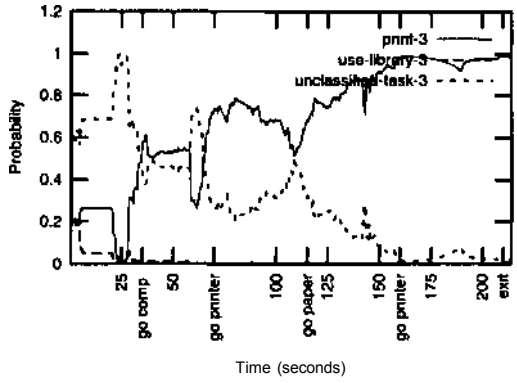

Figure 7: Querying the top level policies

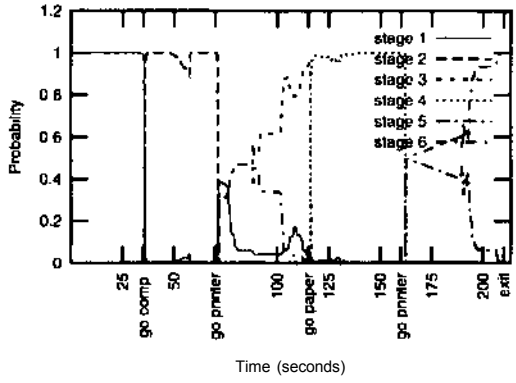

Figure 8: The progress of executing the policy "print'

we have shown that the Rao-Blackwellised Particle Filter (RBPF) approximate inference method used for the AHMM can be extended to the AHMEM, resulting in efficient and scalable procedures for plan recognition in this general setting. Our work demonstrates the advantage of phrasing probabilistic plan recognition as inference in DBN so that suitable approximate methods can be employed to cope with the complexity issue. A similar approach can be applied to more general models to consider more complex plan constructs such as multi-agent plans, interleaving plans etc.

References

[Albrccht *et al,* 1998] David W. Albrecht, Ingrid Zukcrman, and Ann E. Nicholson. Bayesian models for keyhole plan recognition in an adventure game. *User Modelling and User-adapted Interaction,* 8(l-2):5-47, 1998.

[Boutilier *et al,* 1996] Craig Boutilier, Nir Friedman, Moiscs Goldszmidt, and Daphne Koller. Context-specific independence in Bayesian networks. In *Proceedings of the Twelveth Annual Conference on Uncertainty in Artificial Intelligence,* 1996.

iBui *et ai,* 2000] Hung H. Bui, Svetha Venkatcsh, and Geoff West. On the recognition of abstract Markov policies. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-2000),* 2000.

[Bui *et al,* 2002] Hung H. Bui, Svetha Venkatesh, and Geoff West. Policy recognition in the Abstract Hidden Markov Models. *Journal of Artificial Intelligence Research,* 17:451-499, 2002.

[Bui, 2002] Hung H. Bui. Efficient approximate inference for online probabilistic plan recognition. Technical Report 1/2002, School of Computing Science, Curtin University of Technology, Perth, WA, Australia, 2002. Available from http://www.cs.curtin.edu.aurbuihh/.

[Casella and Robert, 1996] George Casella and Christian P. Robert. Rao-Blackwcllisation of sampling schemes. *Hiometrika,* panes 81-94, 1996.

[Charniak and Goldman, 1993] E. Charniak and R. P. Goldman. A Bayesian model of plan recognition. *Artificial Intelligence,* 64:53-79, 1993.

[Doucet *et al,* 2000] Arnaud Doucct, Nando dc Freitas, Kevin Murphy, and Stuart Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence,* 2000.

[Fine *et al,* 1998] Shai Fine, Yoram Singer, and Naftali Tishby. The hierarchical Hidden Markov Model: Analysis and applications. *Machine Learning,* 32, 1998.

iGoldmand *et ai,* 1999] Robert Goldmand, Chistopher Geib, and Chirs Miller. A new model of plan recognition. In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence,* 1999.

[Huber et al, 1994] Marcus J. Huber, Edmund H. Durfee, and Michael P. Wellman. The automated mapping of plans for plan recognition. In *Proceedings of the Tenth Annual Conference on Uncertainly in Artificial Intelligence,* 1994.

[Jelinek et al., 1992] F. Jelinek, J. D. Lafferty, and R. L. Mcrcci. Basic methods of probabilistic context free grammar. In P. Lafaee and R. Dc Mori, editors, *Recent Advances in Speech Recognition and Understanding,* pages 345-360. Springer-Vcrlag, 1992.

[Jensen, 1996] F. Jensen. *An Introduction to Bayesian Networks.* Springer, 1996.

[Murphy and Pashkin, 20011 Kevin Murphy and Mark Pashkin. Linear time inference in hierarchical HMMs. In *NIPS-0l,* 2001.

[Nguyen *et al,* 2003] Nam T. Nguyen, Hung H. Bui, Svetha Venkatesh, and Geoff West. Recognising and monitoring high-level behaviours in complex spatial environments. In *Proceedings of the IEEL International Conference on Computer Vision and Pattern Recognition (CVPR-03),* 2003.

[Nicholson and Brady, 1994] A. E. Nicholson and J. M. Brady. Dynamic belief networks for discrete monitoring. IEEE *Transactions on Systems, Man and Cybernetics,* 24(11): 1593-1610, 1994.

[Parr, 1998] Ronald Parr. *Hierarchical control and learning for Markov Decision Processes.* PhD thesis, University of California, Berkeley, 1998.

[Pynadath and Wellman, 19951 David V. Pynadath and Michael P. Wellman. Accounting for context in plan recognition, with appli cation to traffic monitoring. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence,* 1995.

[Pynadath and Wellman, 2000] David V. Pynadath and Michael P. Wellman. Probabilistic state-dependent grammars for plan recognition. In *Proceedings of the Sixteenth Annual Conference on Uncertainty in Artificial Intelligence,* 2000.

[Pynadath, 1999] David V. Pynadath. *Probablistic grammars for plan recognition.* PhD thesis, Computer Science and Engineering, University of Michigan, 1999.

[Rabiner, 1989] Lawrence R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE,* 77(2):257-286, 1989.