Probabilistic Plan Recognition in Multiagent Systems

Suchi Saria
Computer Science Department
Mount Holyoke College
South Hadley, MA 01075
ssaria@mtholyoke.edu

Sridhar Mahadevan
Department of Computer Science
University of Massachusetts
Amherst, MA 01002
mahadeva@cs.umass.edu

Abstract

We present a theoretical framework for online probabilistic plan recognition in cooperative multiagent systems. Our model extends the Abstract Hidden Markov Model (AHMM) (Bui, Venkatesh, & West 2002), and consists of a hierarchical dynamic Bayes network that allows reasoning about the interaction among multiple cooperating agents. We provide an in-depth analysis of two different policy termination schemes, T_{all} and T_{any} for concurrent action introduced in (Rohanimanesh & Mahadevan 2003). In the T_{all} scheme, a joint policy terminates only when all agents have terminated executing their individual policies. In the T_{any} scheme, a joint policy terminates as soon as any of the agents terminates executing its individual policy. Since exact inference is intractable, we describe an approximate algorithm using Rao-Blackwellized particle filtering. Our approximate inference procedure reduces the complexity from exponential time in N, the number of agents and K, the number of levels, to time linear in both N and $\hat{K} \leq K$ (the lowest-level of plan coordination) for the T_{all} termination scheme and $O(N \log N)$ and linear in $\hat{K}$ for the T_{any} termination scheme.

Introduction

A large number of real-world multiagent domains require reasoning about the team behavior of agents, from sporting events, military and security surveillance, to teams of robots (Mataric 1997) (Marsella *et al.* 1999). Effective coordination is known to be a major challenge in such domains. If agents possess similar abilities and share the same utility function or *common interests*, they run the risk of both pursuing the same objective with the consequence of an undesirable outcome unless they coordinate (Boutilier 1999). For example, in robot soccer, while a team is attacking, say a player is required to block the opponent while the second player receives the ball from its teammate. If both agents position themselves to block the opponent, then the pass fails. Additionally, agents need to coordinate plans at different levels of detail. For example, in executing a pass between two agents, agents need to coordinate at the level of kicking the ball on time and in the right direction such

that the other agent receives the ball without losing it to the opponent team. However, the individual details of muscle movement in each agent is obviously not communicated or coordinated explicitly.

This paper provides a rigorous theoretical framework for representing and reasoning about hierarchical plans in cooperative multiagent systems. Extensive work has been done in opponent model recognition, e.g. (Riley & Veloso 2001) and (Intille & Bobick 1999), and space precludes us from giving a detailed discussion of previous work, but suffice it to say that many earlier approaches are domain-specific. More importantly, much previous work fails to explicitly model hierarchical aspects of coordination among agents. We base our work on (Bui, Venkatesh, & West 2002), which introduced the framework of the abstract hidden Markov Model (AHMM) for plan recognition in single agent systems. We extend their approach and introduce Hierarchical Multiagent Markov Processes as a framework for modeling hierarchical policy execution in multiagent systems. We assume agents coordinate their actions at more abstract levels explicitly using a central controller, but that at lower levels, individual policies are executed without coordination by each agent.

Hierarchical Multiagent Markov Processes

Since we are primarily interested here in plan recognition, and not in finding optimal policies, we do not need the full machinery of Markov decision processes. However, much of our work is inspired by work on multiagent MDP (MMDP) models (e.g., (Boutilier 1999)), although a key weakness of previous models has been the lack of attention paid to policy hierarchies. In particular, our work specifically relies on the notion of a policy hierarchy using ideas from hierarchical reinforcement learning (Barto & Mahadevan 2003; Makar, Mahadevan, & Ghavamzadeh 2001). As in a typical MDP, the world consists of a set of possible states and actions permissible in those states. A policy maps a state to an action (deterministic policy) or a distribution over a set of actions (stochastic policy). Hierarchical policies can invoke more refined policies, i.e., at any state s, a high-level policy π^k is executed by selecting a lower-level policy π^{k-1}, according to the distribution $\sigma_{\pi^k}(s, \cdot)$. The selected policy

π^{k-1} selects among a set of lower-level policies and so on until a primitive action is selected for each agent. Once π_{k-1} terminates in a state d, π^k also terminates with termination distribution $\beta_{\pi^k}(d)$ if d is in the termination set of π^k or continues by selecting another $k-1$ level policy in state d.

The advantage of our approach lies in the way it explicitly models the level of detail to which agents coordinate plans. Below this level, policies are executed independently. To represent this dichotomy in shared execution and individual execution, we specify the lowest coordination level by $\hat{K}$. $\hat{K}$ remains the same throughout the process of policy execution and (for simplicity) is independent of the agents' states. All policies above $\hat{K}$ are shared or *joint* policies. We model joint policies as a MMDP with the exception that they are defined over temporally extended actions for each agent. At level $\hat{K}$, the joint-policy selects a lower level single agent policy for each agent i which further selects lower level policies within the agent's policy hierarchies and so on. To illustrate these ideas through a 5-level example (see Figure 1), consider two agents executing an attack strategy in soccer. One possible way of attack is to execute a multiple-pass to the goal. A multiple-pass involves repeatedly executing a single-pass between two agents at the lower level. Each single pass requires one agent to pass the ball and the second agent to receive the ball. Receiving the ball involves running to the location where the ball is expected which in turn calls a sequence of move primitive actions. This set of recursively selected policies at all levels including the primitive actions defines the policy hierarchy.

Throughout the paper, we use $E_{i,t}^k$ where E is any entity such as a state or a policy. Here, the first subscript i specifies an agent from a set of N agents, the second subscript t specifies the time and the superscript k specifies the level in the hierarchy to which this entity belongs. We use $1{:}n$ to denote the sequence $(1, 2, \cdots, n)$

Definition 1: Local Policy
An example of a local policy μ_i is passing the ball which involves executing a set of actions such as turn, kick and wait. We define the local policy as a tuple $\mu_i = \ll \mathcal{S}_{\mu_i}, \mathcal{D}_{\mu_i}, \beta_{\mu_i}, \sigma_{\mu_i} \gg$ for each agent $i \in \{1, \cdots n\}$ where

- $\mathcal{S}_{\mu_i} \subseteq \mathcal{S}_i \subseteq \mathcal{S}$ is the subset of applicable states for agent i and policy μ_i

- $\mathcal{D}_{\mu_i}$ is the set of termination states

- $\beta_{\mu_i} \colon \mathcal{D}_{\mu_i} \to (0, 1]$ is the termination distribution such that $\beta_{\mu_i}(d) = 1$ for all states $d \in \mathcal{D}_{\mu_i} \setminus \mathcal{S}_{\mu_i}$

- $\sigma_{\mu_i} \colon \mathcal{S}_{\mu_i} \times \mathcal{A}_i \to [0, 1]$ is the action selection function such that for the given policy μ_i and current state s, $\sigma_{\mu_i}(s, a)$ is the probability with which the action a is selected in state s. If the agents are homogeneous, then the set of local policies are the same for all agents and the subscript i can be removed. Each such policy generates a Markov sequence of states defined by the transition model $\sigma_a(s, s')$
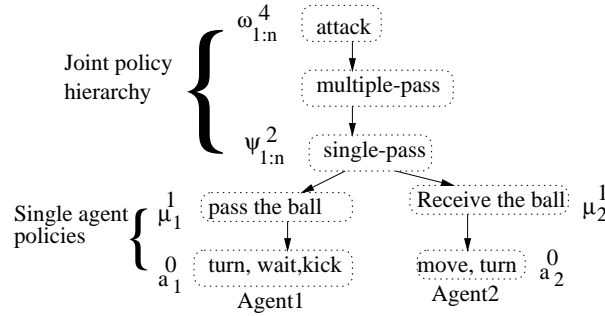

Figure 1: Policy hierarchy illustrating a soccer strategy with two agents.

Definition 2: Abstract Policies
An abstract policy is defined in a way similar to a local policy where states map to other lower-level policies instead of primitive actions.

Definition 2.1 Abstract Single-agent policy over other Single-agent policies
Let Π_i^μ be a set of single agent policies. A tuple $\nu_i = \ll \mathcal{S}_{\nu_i}, \mathcal{D}_{\nu_i}, \beta_{\nu_i}, \sigma_{\nu_i} \gg$ is an abstract policy over other policies in Π_i^μ for each agent $i \in \{1, \cdots, n\}$ where

- $\mathcal{S}_{\nu_i} \subseteq \bigcup_{\mu_i \in \Pi_i^\mu} \mathcal{S}_{\mu_i}$ is the subset of applicable states

- $\mathcal{D}_{\nu_i}$ is the set of termination states

- $\beta_{\nu_i} \colon \mathcal{D}_{\nu_i} \to (0, 1]$ is the termination distribution such that $\beta_{\nu_i}(d) = 1$ for all states $d \in \mathcal{D}_{\nu_i} \setminus \mathcal{S}_{\nu_i}$

- $\sigma_{\nu_i} \colon \mathcal{S}_{\nu_i} \times \Pi_i^\mu \to [0, 1]$ is the mapping of states to abstract policies such that for the given policy ν_i and current state s, $\sigma_{\nu_i}(s, \mu_i)$ is the probability with which the lower level policy μ_i is selected

Definition 2.2 Abstract Joint-agent policy over Abstract Single-agent policies
Performing a single-pass between two agents is an example of an abstract joint policy. It is defined as the tuple $\psi_{1:n} = \ll \mathcal{S}_{\psi_{1:n}}, \mathcal{D}_{\psi_{1:n}}, \beta_{\psi_{1:n}}, \sigma_{\psi_{1:n}} \gg$ over a set of abstract single-agent policies in $(\Pi_1^\nu \times \Pi_2^\nu \cdots \times \Pi_n^\nu)$.

- $\mathcal{S}_{\psi_{1:n}} \subseteq \bigcup_{\nu_1 \in \Pi_1^\nu} \mathcal{S}_{\nu_1} \times \bigcup_{\nu_2 \in \Pi_2^\nu} \mathcal{S}_{\nu_2} \cdots \times \bigcup_{\nu_n \in \Pi_n^\nu} \mathcal{S}_{\nu_n}$ are the applicable states

- $\mathcal{D}_{\psi_{1:n}}$ is the set of termination states

- $\beta_{\psi_{1:n}} \colon \mathcal{D}_{\psi_{1:n}} \to (0, 1]$ is the termination distribution such that $\beta_{\psi_{1:n}}(d_1, d_2 \cdots d_n) = 1$ for all states specified by the tuple $< d_1, \cdots d_n > \in \mathcal{D}_{\psi_{1:n}} \setminus \mathcal{S}_{\psi_{1:n}}$

- $\sigma_{\psi_{1:n}} \colon \mathcal{S}_{\psi_{1:n}} \times (\Pi_1^\nu \times \Pi_2^\nu \cdots \times \Pi_n^\nu) \to [0, 1]$ is the mapping of states to abstract single agent policies such that for the given policy $\psi_{1:n}$ and current state of all agents, $\sigma_{\psi_{1:n}}(s_1, s_2 \cdots s_n, \nu_1, \nu_2 \cdots \nu_n)$ is the probability with which agent 1 executes the policy ν_1, agent 2 executes the policy ν_2 and so on.

Definition 2.3 Abstract Joint-agent policy over other Joint-agent policies
A multiple pass is an example of an abstract joint policy which involves executing a lower-level single-pass joint policy (see Figure 1). An abstract joint policy is defined as

the tuple $\omega_{1:n} = \ll \mathcal{S}_{\omega_{1:n}}, \mathcal{D}_{\omega_{1:n}}, \beta_{\omega_{1:n}}, \sigma_{\omega_{1:n}} \gg$ over a set lower level abstract joint-policies in $\Pi_{1:n}^{\psi}$ for all agents where

- $\mathcal{S}_{\omega_{1:n}} \subseteq \bigcup_{\psi \in \Pi_{1:n}^{\psi}} \mathcal{S}_{\psi_{1:n}}$ is the set of applicable states

- $\mathcal{D}_{\omega_{1:n}}$ is the set of termination states

- $\beta_{\omega_{1:n}} : \mathcal{D}_{\omega_{1:n}} \to (0,1]$ is the termination distribution such that $\beta_{\omega_{1:n}}(d) = 1$ for all states $d \in \mathcal{D}_{\omega_{1:n}} \setminus \mathcal{S}_{\omega_{1:n}}$

- $\sigma_{\omega_{1:n}} : \mathcal{S}_{\omega_{1:n}} \times \Pi_{1:n}^{\psi} \to [0,1]$ is the mapping of states to abstract joint-policies such that for the given joint-policy $\omega_{1:n}$ and current state $s_{1:n}$ of all agents $\sigma_{\omega_{1:n}}(s_{1:n}, \psi_{1:n})$ is the probability with which the lower-level joint policy $\psi_{1:n}$ is selected

Definition 3: Policy Hierarchy A policy hierarchy is the set of recursively defined policies at all levels including the primitive actions. It is represented by the tuple, $\overline{H} = \{(\Pi_1^0, \Pi_2^0 \cdots \Pi_n^0), \cdots (\Pi_1^{\hat{K}-1}, \Pi_2^{\hat{K}-1} \cdots \Pi_n^{\hat{K}-1}), \Pi_{1:n}^{\hat{K}}, \cdots \Pi_{1:n}^{K}\}$ where the lowest-level joint policy is defined at level $\hat{K}$.

In the soccer example in Figure 1, $\hat{K} = 2$. The policy at level $\hat{K}$ calls a level $(\hat{K} - 1)$ single agent policy for each agent, each of which call other level $(\hat{K} - 2)$ policies in the single-agent policy hierarchies and so on. $(\Pi_1^0, \Pi_2^0 \cdots \Pi_n^0)$ is the tuple representing the set of primitive actions for all agents. Even though the definition above represents a balanced policy hierarchy, it is possible to specify an unbalanced hierarchy by introducing dummy policies at the higher levels which are the same as its lower-level policies or primitive actions. The cardinality of the joint policy at level $\hat{K}$ is $|\Pi_1^{\hat{K}-1}| \times |\Pi_2^{\hat{K}-1}| \cdots \times |\Pi_n^{\hat{K}-1}|$. Although complete coordination can be modeled by defining policies at all levels including primitive actions as joint policy nodes, this causes the number of possible joint-policies to blow up. Specifying the joint policies only at levels $\hat{K} > 1$ allows the agents to abstract away individual lower level planning details from other agents and reduces the complexity of inference in the network.

Dynamic Bayesian Network representation

We view the policy recognition problem in a multi-agent system as probabilistic inference on a dynamic Bayes Network (DBN). To explain the full DBN, we first describe the two fundamental sub-structures of policy termination and policy selection for each of the two termination mechanisms, T_{any} and T_{all}, introduced in the concurrent action model (CAM) (Rohanimanesh & Mahadevan 2003). We construct the full network by "stacking up" these sub-structures for K layers just as in the AHMM (Bui, Venkatesh, & West 2002). In each sub-structure, s_{t-1} represents the relevant state variable at time $t - 1$. Let π_t^k represent the policy variable at time t and level k. Single agent policy nodes within the agent's policy hierarchy, i.e., at level $l < \hat{K}$, just depend on the state of that agent. The joint-policy nodes depend on $s_{1:n,t-1}$, the states of all agents at time $t - 1$. The natural

termination of policy π_t^k is represented by the boolean variable e_t^k, which becomes true when the policy π_t^k terminates in a state d according to the termination distribution $\beta_{\pi_t^k}(d)$. An additional coordination node γ_{t-1}, a boolean variable, is defined at level $\hat{K}$ which controls influence on a single agent policy hierarchy from other single agent policy hierarchies under the pre-defined termination scheme.

1. Policy termination Due to the way the model is defined, within an agent's individual policy hierarchy and within the joint policy hierarchy, policies cannot *naturally* terminate if a lower level policy is still continuing its execution. In other words, in the context of $e_t^{k-1} = F$ (false), e_t^k deterministically assumes the value false and is independent of both the state and policy. Following the notion of context-specific independence (CSI) (Boutilier *et al.* 1996), as shown in the graphical representation in Figure 2, the links from the policy and states to the termination nodes can be removed. If $e_t^{k-1} = T$ (true), then $e_t^k = T$ with probability $\beta_{\pi}(s_t)$.
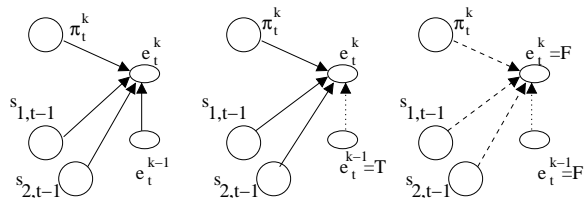

Figure 2: Sub-structure for natural policy termination. Dashed arcs represents context-specific independence.

2. Policy selection For policy selection, the current policy π_t^k depends on the higher level policy π_t^{k+1}, the state variable at previous time step s_{t-1}, its own policy value π_{t-1}^k, and its termination status e_{t-1}^k. As shown in Figure 3, e_{t-1}^k serves as the context variable that controls the dependency of π_t^k on its parents. If the previous policy has not terminated, then the current policy is the same as the previous policy and as shown using CSI, is independent of both the state and higher level policy. If the previous policy has terminated, i.e., $e_{t-1}^k = T$, then a new policy is selected independent of the previous policy from the distribution $P(\pi_t^k | \pi_t^{k+1}, s_{t-1})$ based on the previous state and the higher level policy. For levels below $\hat{K}$, the policy selection sub-structure has additional dependency on the joint-termination node, γ_{t-1}. When $\gamma_{t-1} = T$, even if the policies in the single agent hierarchy have not terminated at time $t - 1$, new policies are selected instead at time t. This is called *interruption* or *forced policy selection*.

- In the T_{all} termination scheme, γ_{t-1} becomes true only when all individual policy termination nodes at level $\hat{K} - 1$ and time $t - 1$ become true, i.e., all agents have naturally terminated their individual policy hierarchies. For example, while coordinating to execute a single pass
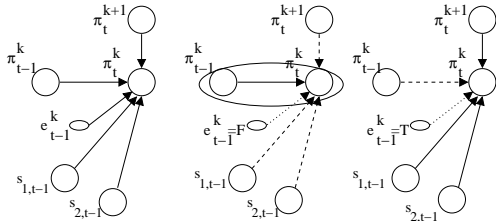
Figure 3: General sub-structure for policy selection. Dashed arcs represent context-specific independence relations.

strategy, γ_{t-1} becomes true only when both agents successfully finish their individual policies, i.e., Agent 1 has successfully made the pass and Agent 2 has successfully received the ball. When not all agents have terminated, agents that have terminated, independent of the state and previous policy, repeatedly execute a wait or *one-step no-op* policy until all agents terminate.

- In the T_{any} termination scheme, γ_{t-1} becomes true if any agent from the set of agents terminates at time $t - 1$, otherwise it remains false. Consider the example, where two teammates are running to chase the ball. The agents stop chasing the ball as soon as any one of the teammates acquires the ball. Now a new policy is selected for both teammates even though only one member naturally terminated. Hence, when $\gamma_{t-1} = T$, even if $e_{t-1} = F$, the remaining agents that did not terminate naturally are forced to select a new policy as shown in Figure 4. In the context that $\gamma_{t-1} = F$, policy selection at time t takes place in the same way as shown in Figure 3.
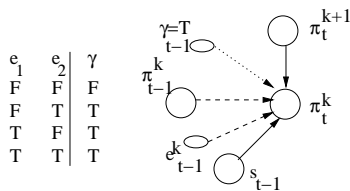

Figure 4: Policy selection under T_{any} termination for two agents. Dashed arcs represent context-specific independence relations.

3. The full DBN structure The DBN shown in Figure 5 can be constructed by super-imposing the above sub-structures of policy termination and policy selection at each level. To get an intuitive understanding of the DBN, note that each agent's individual policy hierarchy is an AHMM with $\hat{K} - 1$ policy levels. All single agent AHMMs are connected through γ_{t-1} which controls influence on a single agent policy hierarchy from the higher level joint policy nodes as well as other single agent policy hierarchies. γ_{t-1} becomes true under the pre-specified termination scheme when agents terminate at the highest level of policy execution in their AHMM. The levels at and above $\hat{K}$ are modeled just as those in the AHMM except the policies are defined

as joint policies which depend on the states of all agents. It is easy to see how the network can be extended to multiple agents by extending the joint policy at and above level $\hat{K}$. To model complete coordination, the model simplifies to look like a single agent AHMM with the exception that the policy and action nodes are defined as joint nodes over the states of all agents. Additionally, the joint-termination node γ is removed because it always assumes the value true since actions terminate naturally at each time step. It is also possible to model coordination among groups of agents where each group can specify a degree of coordination, i.e., the level at which their joint policy is specified independently. The individual subgroup's joint-policies are now treated as single agent policies in the larger DBN.
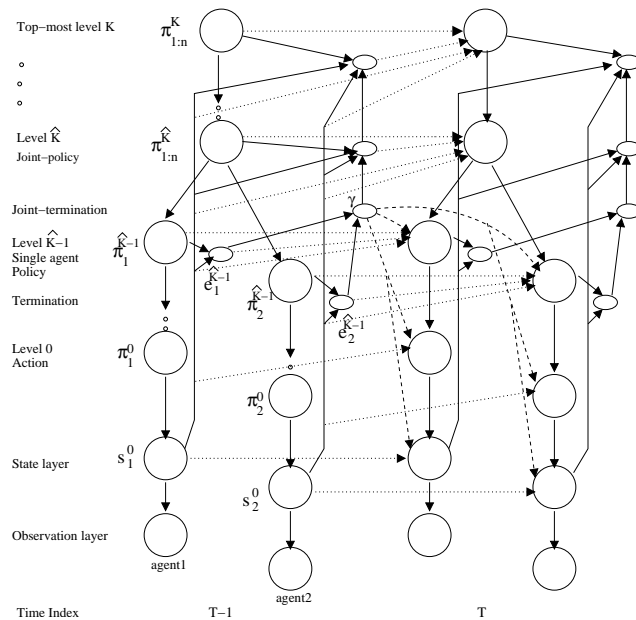

Figure 5: The DBN representation showing two agents. Each agent policy hierarchy is an AHMM with $\hat{K}$ levels. γ at level $\hat{K} - 1$ is the coordination node. In the T_{all} scheme, the dashed arcs from γ_t to the single agent policies below level $\hat{K} - 1$ are removed. The dotted-arcs represents the inter-time slice dependency.

4. Conditional Independence Properties in a Time Slice of the DBN We had presented the problem of plan recognition as probabilistic inference on the DBN shown in Figure 5. The belief state of our DBN representing the execution of HMMP is a joint distribution over $N\hat{K}$ single agent policy nodes, N states, N highest level of termination values for all N agents and $K - \hat{K} + 1$ joint policy nodes. Thus, generally the size of the belief state representation is exponential in the size of K and N and there is no compact way to represent the belief state. This makes exact inference in the network intractable for large K and N. However, both the policy selection and

the policy termination sub-structures discussed earlier motivate additional conditional independence statements that simplify the structure of a time slice in the network. In both sub-structures, termination nodes e_{t-1}^k and γ_{t-1} serve as the context variables, i.e., knowing the termination variables simplifies the network using the notion of Context Specific Independence. At the time of policy selection, policies above level k influence the policies below level k only through the policy at level k. Hence, by conditioning on the starting state and the starting time of the policies, we can exploit additional conditional independence structure in the network. We generalize the conditional independence theorem in (Bui, Venkatesh, & West 2002) to express the conditional independence properties in our network.

Conditional Independence Theorem *Given the policy at level k, its starting time and starting state, all policies and states below level k are independent of the policies above level k.*

We now re-state and discuss this theorem for two special cases in our network.

- **Case 1: At level $k \geq \hat{K}$.**
  Let $\tau_t^k = \max\{t' < t | e_{t'}^k = T\}$ be the random variable representing the starting time of the current level k policy $\pi_{1:n,t}^k$. Let $\alpha_{1:n,t}^k$, the state at time τ_t^k, be its starting state. Let $\pi_{1:n,t}^{<k} = \{s_{1:n,t}, \pi_{1,t}^{1:\hat{K}-1}, \pi_{2,t}^{1:\hat{K}-1} \cdots \pi_{n,t}^{1:\hat{K}-1}, \pi_{1:n,t}^{\hat{K}} \cdots \pi_{1:n,t}^{k-1}\}$ denote the set of all single agent policies, joint policies and states below level k at the current time t and $\pi_{1:n,t}^{>k} = \{\pi_{1:n,t}^{k+1}, \cdots \pi_{1:n,t}^K\}$ denote the set of current policies above level k. Then, given the current policy $\pi_{1:n,t}^k$ at level k, its starting state and starting time, the set of all single agent policies, states and joint policies below level k are independent of the set of all joint policies above level k. This is written as:

$$\pi_{1:n,t}^{<k} \perp \pi_{1:n,t}^{>k} \mid \pi_{1:n,t}^k, \alpha_{1:n,t}^k, \tau_t^k \qquad (1)$$

  Enumerating all the agent states, and treating the joint policies as the set of basic policies, the layers above k are same as those in the AHMM. Using context specific independence properties described earlier in this section, the proof of the AHMM can be directly extended to this general case of joint policies.

- **Case 2: At level $k < \hat{K}$.**
  Let $\tau_t^k = \max\{t' < t | \gamma_{t'} = T \text{ or } e_{t'}^k = T\}$ be the random variable representing the starting time of the current level k policy $\pi_{i,t}^k$. Let $\alpha_{i,t}^k$, the state at time τ_t^k, be its starting state. Let $\pi_{i,t}^{<k} = \{s_{i,t}, \pi_{i,t}^{1:k-1}\}$ denote the set of single agent policies below level k and the state for agent i at time t. Then, given the current policy $\pi_{i,t}^k$ at level k, its stating state and time, the set of lower level policies and state in the policy hierarchy for agent i are independent of the set of all other policies at time t, including the joint policies and other agent's individual policy hierarchies.

This is written as:

$$\pi_{i,t}^{<k} \perp \pi_{i,t}^{(k+1):\hat{K}-1}, \pi_{1:n,t}^{\hat{K}:K}, \forall j_{j\neq i} \pi_{j,t}^{0:\hat{K}-1},$$

$$\forall j_{j\neq i} s_{j,t} \mid \pi_{i,t}^k, \alpha_{i,t}^k, \tau_t^k \qquad (2)$$

Proof sketch: Each single agent hierarchy is modeled as an AHMM. Policy selection at any level k below $\hat{K}$ in our network has additional dependency on the joint termination γ_{t-1}. Hence, conditioning on γ_{t-1} as well leads to the same CSI properties as discussed for the AHMM and $\pi_{i,t}^k, \alpha_{i,t}^k,$ and τ_t^k d-separate $\pi_{i,t}^{<k}$ from the rest of the variables in the time-slice.

Inference in the network

The complexity of inference in the DBN depends on the size of representation of the belief state. In general, in our DBN, the belief state we need to maintain does not preserve the conditional independence properties of the single time-slice network discussed above, making exact inference intractable even when the DBN has a sparse structure. The belief state for our model is the set of all variables in the current time slice t conditioned on the observation o_t written as:

$$P(\pi_{1,t}^{<\hat{K}}, \pi_{2,t}^{<\hat{K}}, \cdots \pi_{n,t}^{<\hat{K}}, \pi_{1:n,t}^{\hat{K}}, \cdots, \pi_{1:n,t}^K, s_{1:n,t}, e_{1,t}^{1:\hat{K}-1}, \cdots$$

$$\cdots, e_{n,t}^{1:\hat{K}-1}, e_{1:n,t}^{\hat{K}} \cdots, e_{1:n,t}^K, \gamma_t \mid o_{1:n,t}) \qquad (3)$$

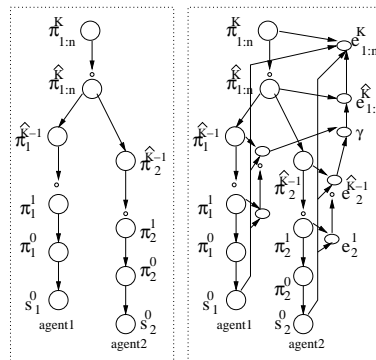Exact inference is clearly not scalable to a large domain

Figure 6: The left structure is the belief tree T_t. The belief state $\mathcal{B}_t$ is obtained by attaching the termination nodes to T_t. The tree has its root at the $\pi_{1:n,t}^K$

where several agents are interacting. However, if we assume: 1) the state sequence can be observed with certainty 2) the exact time when each policy starts and ends is known, then the Conditional Independence Theorem stated earlier holds and as a direct consequence of this theorem, the belief state decomposes into the simple tree-like structure shown in Figure 6. The new belief state is now conditioned on both the state as well as termination node values upto time $t-1$ since they are now observed:

$$P(e_t^{all}, \pi_t^{all}, s_{1:n,t}|s_{1:n,0:t-1}, e_{1:t-1}^{all})$$

$$= P(e_t^{all}|\pi_t^{all}, s_{1:n,t})P(\pi_t^{all}, s_{1:n,t}|s_{1:n,0:t-1}, e_{1:t-1}^{all})$$

$$= P(e_t^{all}|\pi_t^{all}, s_{1:n,t})T_t$$

The new belief state is realized by adding the links from the current policies and current states to the terminating nodes in T_t as shown in Figure 6. On analyzing the size of our belief state, we note that each node has a manageable size. The domain for a joint policy variable π_t^k is Π^k, the set of all policies at level k. If π_t^k is a single agent policy, then its domain is further limited to Π_i^k, the set of policies at level k only defined in the policy hierarchy of agent i. Given the starting state α_t^k of a policy, the set of possible policies is limited to $\pi_t^k \in \Pi^k(\alpha_t^k)$ and is independent of K. Similarly, the domain for s_t, the state at time t is the set of states reachable from s_{t-1} in one primitive action. More generally, if $\mathcal{N}$ is the maximum number of of relevant neighboring states and policies at any single level of the network, then for any link, the conditional probability table is $\mathcal{O}(\mathcal{N}^2)$ and the overall size of the belief chain is $\mathcal{O}(L\mathcal{N}^2)$ where $L = \hat{K}N + (K - \hat{K})$ is the number of links in the belief tree T_t.

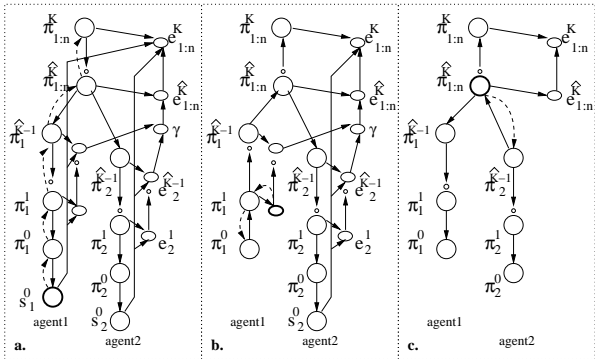**Exact-step:** We now briefly describe an algorithm that

Figure 7: Representation of the belief state update from $\mathcal{B}_t$ to $\mathcal{B}_{t+}$.The root of the tree is shown as the bolded node. The dashed arcs show the resulting arcs after arc-reversal at that step

recursively updates the simple belief state shown in Figure 6 in closed form. It is possible to use the general junction tree algorithm (Lauritzen & Spiegelhalter 1988) instead to update the belief state, but at the expense of converting from the undirected graph to a directed graph after inference at each time-step to allow efficient sampling from the network. Given the complete specification of the belief state $\mathcal{B}_t$, the new belief state $\mathcal{B}_{t+1}$ can be computed by 'rolling-over' $\mathcal{B}_t$ (Boyen & Koller 1995) using the following steps:

1) Absorbing the new evidence: Here, we instantiate all the state and termination nodes at time t, to obtain $\mathcal{B}_{t+}$. We maintain the belief tree T_t with its root at the highest level of termination in the network. When we say that the tree T_t has root at a node, it means that all links in the tree point away from that node. To move the root from a node

at level k to any level say k', we iteratively reverse the link between adjacent nodes starting at k till we reach k' using the standard link-reversal operation (Shachter 1986). We start by instantiating the state and termination nodes in the single agent hierarchies as described for the AHMM. To do this for agent 1, as shown in Figure 7a, the root is first moved to $s_{1,t}$ to instantiate the state for agent 1. Once a node has been instantiated and absorbed, it is removed from the belief state. We then instantiate $e_{1,t}^{1:\hat{K}-1}$, the termination nodes in agent 1's policy hierarchy starting with $e_{1,t}^1$. Within the single agent policy hierarchies, a policy at a higher level cannot *naturally* terminate if any policy at the lower level has not terminated. Hence, the only valid instantiation of the termination nodes is such that $l_{1,t} = \{k'|k \in 1, \cdots, \hat{K} - 1; \forall k > k', e_{1,t}^k = F; \forall k < k', e_{1,t}^k = T\}$ where $l_{1,t}$ is the highest level of termination within the single agent hierarchy of agent 1. All policies below $l_{i,t}$ must terminate and all policies above must not terminate. To absorb each instantiated termination node, we iteratively reverse the links from $\pi_{i,t}^{k-1}$ to $\pi_{i,t}^k$ and $\pi_{i,t}^k$ to $e_{i,t}^k$ for agent i as shown in Figure 7b. We repeat this process of instantiating the state and termination nodes for every agent's individual policy hierarchy. The last link reversed while repeating this process for every agent policy hierarchy is from $\pi_{i,t}^{\hat{K}-1}$ to $\pi_{1:n,t}^{\hat{K}}$. The resulting belief state as shown in Figure 7c after all the agent's states and termination nodes in the single agent policy hierarchies have been instantiated has its root at level $\hat{K}$.

Now the remaining termination nodes $e_{i,t}^{\geq \hat{K}}$ are instantiated in the same way by reversing the link from $\pi_{1:n,t}^{k-1}$ to $\pi_{1:n,t}^k$ and $\pi_{1:n,t}^k$ to $e_{1:n,t}^k$ iff the joint-termination $\gamma_t = T$ (A longer version of the paper contains details on this procedure). The termination nodes are instantiated only until $e_{1:n,t}^k = F$, i.e., till a joint-policy at level k terminates. As discussed earlier, all policies above level k by default do not terminate because the policy at level k has not terminated. The resulting belief tree after all the states and termination nodes have been instantiated, has its root at $\pi_{1:n,t}^{r_t}$ where r_t is at $\hat{K}$ by default if no joint policies terminate or at the highest level k where a joint policy has terminated.

2) Projecting the belief state into the next step: This step creates the new belief tree T_{t+1} from $\mathcal{B}_{t+}$. Since the policies at and above r_t have not terminated, the marginals are retained in the belief tree T_{t+1} as shown in Figure 8a. For updating the belief tree, the parameters of the new subtree starting at $k \leq r_t$ are obtained from one of the following policy selection distributions:

- $\sigma_{\pi_{1:n,t+1}^{k+1}}(s_{1:n,t}, \pi_{1:n,t+1}^k), k \geq \hat{K}$

- $\sigma_{\pi_{1:n,t+1}^{k+1}}(s_{i,t}, \pi_{i,t+1}^k), k = \hat{K} - 1$

- $\sigma_{\pi_{i,t+1}^{k+1}}(s_{i,t}, \pi_{i,t+1}^k), k < \hat{K} - 1$

Let $l_{i,t} \in \{1 \cdots \hat{K} - 1\}$ be the highest level of termination for each agent in the single agent policy hierarchy. If none of the joint policies have terminated as shown in
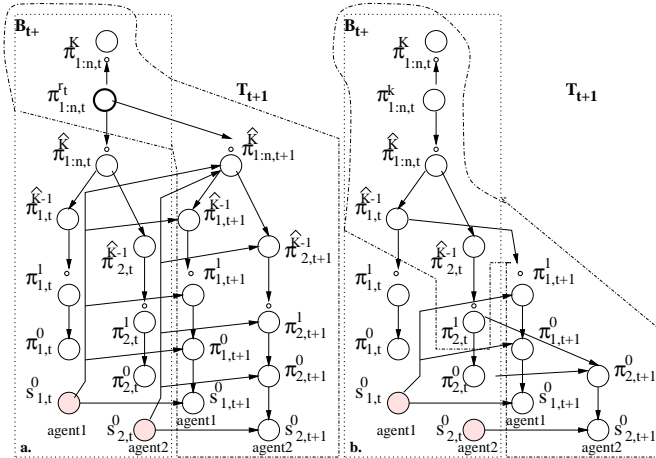
Figure 8: Belief state updating from $\mathcal{B}_{t+}$ to T_{t+1}. All marginals of $\mathcal{B}_{t+}$ above the highest level of termination are retained in T_{t+1} a. shows update procedure when the highest level termination is $\geq \hat{K}$. b. shows update procedure when no joint policies have terminated. Here agent1 has its highest level of termination at level $\hat{K}-2$ and agent2 at level 0.

Figure 8b, then for each agent i, all policies above $l_{i,t}$ are retained as well in T_{t+1} and a new subtree is formed for remaining levels at and below level $l_{i,t}$ using the policy selection distribution given above. The new belief tree is a combination of the belief tree T_t and newly created sub-tree. The new belief state $\mathcal{B}_{t+1}$ is obtained by adding the termination variables to T_{t+1} as shown in Figure 6. This ends the exact update procedure for our network.

The complexity of the belief update procedure is proportional to N, the number of agents and r_t, the level at which the root of the tree is maintained, because the algorithm only needs to modify the bottom r_t levels. The probability that a single agent policy terminates at level k is assumed to be exponentially small w.r.t. k (Bui, Venkatesh, & West 2002). Consequently, for the T_{all} termination scheme, the probability that the highest level of termination in a given time step is at any level k is exponentially small with respect to N and k. The expected value of the highest level of termination at each time step is $\mathcal{O}(\sum_k (\frac{\mathbf{e}-1}{\mathbf{e}})^N \frac{l}{\mathbf{e}^{Nk}})$ which is constant bounded in K, the total number of levels in the network and exponentially decreasing in N. $\mathbf{e}$ represents the base of the natural log. Since, the root of the belief tree by default is maintained at $\hat{K}$, the average update complexity at each time step for the T_{all} termination scheme becomes $\mathcal{O}(N\hat{K})$.

For the T_{any} termination scheme, probability that the highest level of policy termination is at any level L is calculated as follows. Let l_i be the highest level of termination for each agent i.

$$Pr(\max_{i=1}^N l_i = L)$$

$$= Pr(\max_{i=1}^N l_i \leq L) - Pr(\max_{i=1}^N l_i \leq L-1)$$

$$Pr(\max_{i=1}^N l_i \leq L)$$

$$= \prod_N Pr(l_i \leq L)$$

$$= \prod_N (\frac{\mathbf{e}-1}{\mathbf{e}})(1 + \frac{1}{\mathbf{e}} + \frac{1}{\mathbf{e}^2} \cdots + \frac{1}{\mathbf{e}^L})$$

$$= (1 - \frac{1}{\mathbf{e}^{L+1}})^N$$

$$Pr(\max_{i=1}^N l_i = L)$$

$$= (1 - \frac{1}{\mathbf{e}^{L+1}})^N - (1 - \frac{1}{\mathbf{e}^L})^N$$

Thus, the expected value for the highest level of termination at each time step is

$$O(\sum_k k((1 - \frac{1}{\mathbf{e}^{k+1}})^N - (1 - \frac{1}{\mathbf{e}^k})^N)) \qquad (4)$$

We analyze the complexity graphically by plotting the expected value (eq. 4) for plan hierarchies with varying number of agents ($1 - 1000$ agents) and varying number of levels ($1 - 500$ levels) as shown in Figure 9. It is apparent from the plot that the expected value is independent in the total number of levels in the plan hierarchy. Also, we superimpose the plot for the logarithm of n for each l and offset it by 0.5 for clarity. The expected value clearly varies logarithmically in n as shown. Hence, the average update complexity at each time step for T_{any} is $O(\max(NlogN, N\hat{K}))$

Approximation step: The above algorithm is used to update the belief state only if both assumptions stated earlier hold, i.e., the agent states as well as the time when a policy starts and terminates is known. In any real world application, these assumptions are too restrictive. We had seen earlier that if these assumptions do not hold then the belief state no longer has the simple tree-like structure. Our goal is to estimate the marginal $P(\pi_{t+1}^k|o_t)$. One possible approach is to sample from the network to calculate this marginal (Doucet, Godsill, & Andrieu 2000). However, sampling in the product space of all variables in a given time slice of the network becomes less effi cient and accurate with large K and N.
To improve the accuracy of sampling in the network, we use Rao-Blackwellized Particle Filtering (Doucet *et al.* 2000) to analytically marginalize some of the variables and sample only the remaining variables in the belief state. As a consequence of the Rao-Blackwell theorem (Casella & Robert 1996) stated below, the Rao-Blackwellized estimator is generally more accurate than any sampling estimator that involves sampling all variables for the same number of samples N.

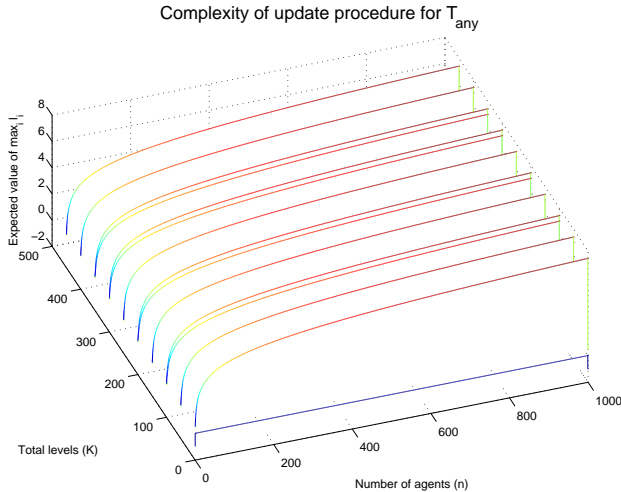$$Var(U) = Var(E[U|V]) + E[Var(U|V)]$$

Figure 9: This plot shows empirically that the expected value of the highest level of termination under T_{any} is independent of the number of levels and is bounded logarithmically in the number of agents.

where U is the set of all variables and V is the set of sampled variables. Hence $Var(U) \geq Var(E[U|V])$. We use our context variables, the states and termination nodes as our RB variables. Conditioning on these variables yields a lower variance estimator for the marginals π_t^{all} and simplifies the network to the simple tree-like structure. Now, we can use the exact-step described above to update the belief state once the RB variables have been sampled. Applying RBPF to our network is tricky because the sequence of RB variables that we are using do not satisfy the Markov property. To sample efficiently from the network, for each agent i, we first position the root of T_t at $s_{i,t}$ and reverse the link from $o_{i,t}$ using evidence reversal (Kanazawa, Koller, & Russell 1995). This gives us the network structure with $o_{i,t}$ absorbed into the network. Now, we perform forward sampling starting from the root node and proceeding upward to sample $s_{i,t}$ and $e_{i,t}$. The joint policy termination nodes are sampled after all the single agent policy termination nodes have been sampled. The samples of the policy nodes are discarded since they are not needed. Sampling is stopped at the first level k in the single agent policy hierarchy of agent i where $e_{i,t} = F$. Similarly, the joint policy termination nodes are only sampled if $\gamma_t = T$. If M samples are maintained, then the overall complexity of maintaining the samples on average at each time step for T_{all} is $\mathcal{O}(MN)$ and is constant bounded in K, the number of levels in the network. For T_{any}, the average sampling complexity at each time step for is $\mathcal{O}(M \max(NlgN, N\hat{K}))$. In the limit, $M \rightarrow \infty$, the above inference algorithm performs as well as exact inference.

For t = 0,1 ...
   For each sample i = 1,...M

Sample $s_{1:n,t}^{(i)}, e_t^{all(i)}$ from $\mathcal{B}_t^{(i)}(s_{1:n,t}^{(i)}, e_t^{all(i)}|o_t)$
Update weight $w^{(i)} = w^{(i)}\mathcal{B}_t^{(i)}(o_t)$
Compute the posterior RB belief state
   $\mathcal{B}_{t+}^{(i)} = \mathcal{B}_t^{(i)}(\pi_t^{all}|s_{1:n,t}^{(i)}, e_t^{all(i)}, o_t)$
Compute the belief tree $T_{t+1}^{(i)}$ from $\mathcal{B}_{t+}^{(i)}$
Compute the new belief state $\mathcal{B}_{t+1}^{(i)}$ from $T_{t+1}^{(i)}$
Compute the marginal $h_{\pi_{t+1}^k}^{(i)} = T_{t+1}^{(i)}(\pi_{t+1}^k)$
   End
   Compute the estimator $P(\pi_{t+1}^k|o_t) = \sum_{i=1}^{M} h_{\pi_{t+1}^k}^{(i)} w^{(i)}$
End

Experimental Results

We present here an application of the HMMP framework to the problem of recognizing behavior of two agents in a simulated domain of our lab. The coordinates of the agents in the lab can be obtained by laser tracking by the robot. Our DBN has a 4 level action/policy hierarchy (see Figure 10). The lowest coordination level is defined at $\hat{K} = 2$. We define the parameters of the policies manually to simulate the movement of the agents in the lab. To represent the uncertainty in our observations, we assume that the agents can be anywhere among its two neighboring states with probabilities defined by a pre-specified model. For a typical sample trajectory, exact inference in the network returns the probabilities of the joint policies and the single agent policies. Shown below is an *observed* sample trajectory for when both agents are trying to exit separately from the two opposite doors in the lab under the T_{all} termination scheme. As the observations about the trajectories arrive over time, the predicted probability distribution for the agents' exit policy, the highest level goal in the network, using exact inference is shown in Figure 11. To show that our approximate inference algorithm can perform as well as exact inference for a large enough sample set, we compute the same probability distribution using approximate inference with 500 samples and get similar results (see Figure 12).

Trajectory of Agent 1: 3,2,3,2,4,5,6,6,8,8,9,7,8,9,7
Trajectory of Agent 2: 6,6,5,5,3,4,5,4,3,3,1,2,1,1,2

In the network for T_{any}, only at level 3, we define new policies such as go to the left door when we hear a knock, and answer the telephone when the phone rings. Here is a sample trajectory where both agents proceed to the left door when a knock is heard at time step 1. At time step 10, the telephone rings and agent 1 proceeds to answer the telephone while agent 2 is still at the left door. The predicted marginal distributions for the highest level goal node using exact and approximate inference are shown in Figure 13 and Figure 14 respectively.

Trajectory of Agent 1: 8,8,7,6,5,6,5,4,4,4,5,6,6,7,8
Trajectory of Agent 2: 7,6,5,5,4,3,2,3,2,1,1,1,1,1,1

Conclusions and Future Work

We have presented Hierarchical Multiagent Markov Processes(HMMP) as a framework for hierarchical probabilis-
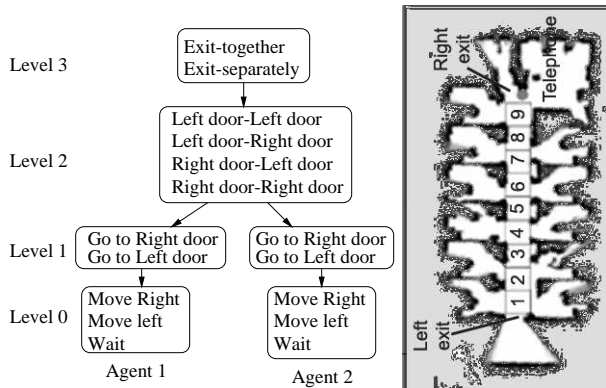
Figure 10: The policy hierarchy for the T_{all} termination. 2) The ALL Lab and states as marked.
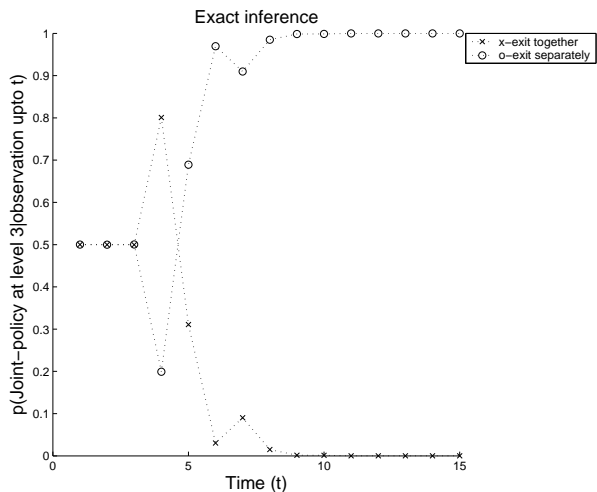

Figure 11: Exact inference results showing the evolution of the probability of the highest level policy under T_{all} termination.
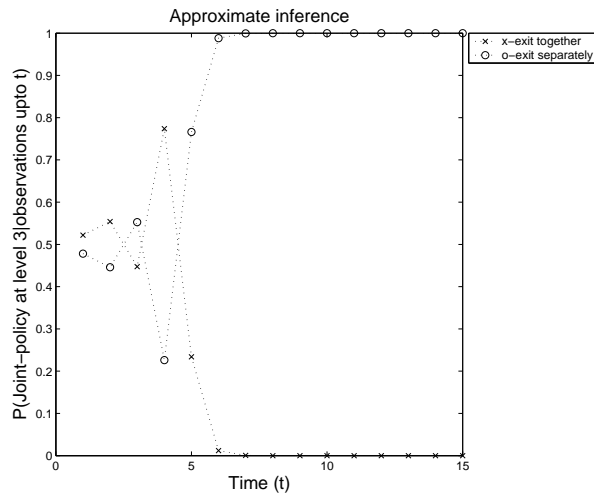

Figure 12: Approximate inference results showing the evolution of the probability of the highest level policy under T_{all} termination.
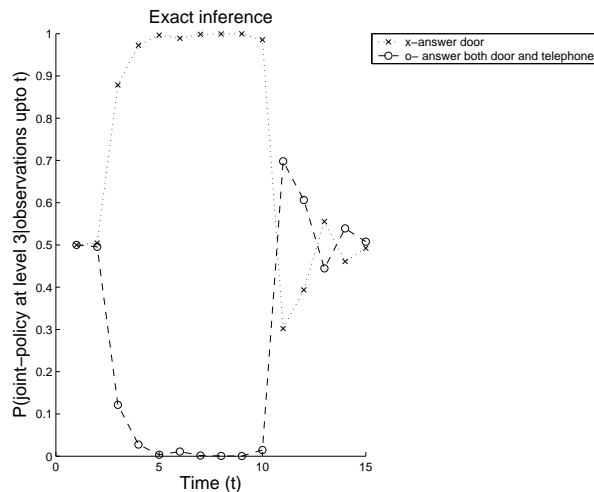
Figure 13: Exact Inference results showing the evolution of the probability of the highest level policy under T_{any} termination

tic plan recognition in cooperative multiagent systems. This framework extends the Abstract Hidden Markov Model to multiagent systems. We analyzed in detail the process of inference in our network for two coordination mechanisms: T_{any} and T_{all}. We showed that using an efficient sampling scheme and exploiting additional conditional independence relations in the network, the Rao-Blackwellized Particle Filter approximate inference method greatly reduces the complexity of the inference in the network.

We briefly outline some directions for future work. One interesting direction to pursue is to investigate techniques for adapting the parameters of the hierarchical DBN from example trajectories. Adaptation would be a useful component of many real-world multiagent plan recognition systems, e.g. robot soccer, where the capacity to adapt to the opponents strategy may prove invaluable. Another interesting direction is to investigate other forms of combining individual agent policies, including probabilistic methods such as "noisy-T-any" and "noisy-T-all". Finally, we plan to apply the proposed framework to a real-world multiagent task, to test its scaleability and effectiveness.

Acknowledgements

References

Barto, A., and Mahadevan, S. 2003. Recent advances in hierarchical reinforcement learning. *Special Issue on Reinforcement Learning, Discrete Event Systems journal* 13:41–77.

Figure 14: Approximate inference results showing the evolution of the probability of the highest level policy under T_{any} termination.
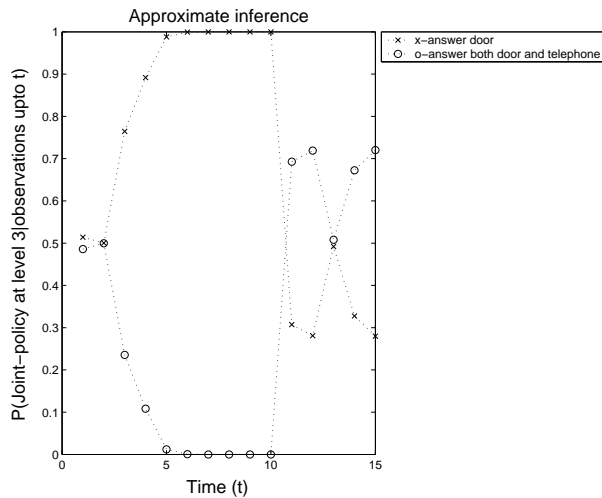
Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context-specific independence in Bayesian networks. In *Proceedings of Uncertainty in Artificial Intelligence*, 115–123.

Boutilier, C. 1999. Sequential optimality and coordination in multiagent systems. In *International Joint Conference in Artificial Intelligence*, 478–485.

Boyen, X., and Koller, D. 1995. Tractable inference for complex stochastic processes. In *Proceedings of Uncertainty in Artificial Intelligence*, 33–42.

Bui, H.; Venkatesh, S.; and West, G. 2002. Policy recognition in the Abstract Hidden Markov Model. *Journal of Artificial Intelligence Research* 17:451–499.

Casella, G., and Robert, C. 1996. Rao-Blackwellization of sampling schemes. *Biometrika* 83:81–94.

Doucet, A.; de Freitas, N.; Murphy, K.; and Russell, S. 2000. Rao-Blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of Uncertainty in Artificial Intelligence*, 176–183.

Doucet, A.; Godsill, S.; and Andrieu, C. 2000. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing* 10(3):197–208.

Intille, S., and Bobick, A. 1999. A framework for recognizing multi-agent action from visual evidence. In *National Conference for Artificial Intelligence*.

Kanazawa, K.; Koller, D.; and Russell, S. 1995. Stochastic simulation algorithms for dynamic probabilistic networks. In *Proceedings of Uncertainty in Artificial Intelligence*, 346–351.

Lauritzen, S., and Spiegelhalter, D. 1988. Local computations with probabilties on graphical structures and their application to expert systems. *Royal Statistical Society B* 50:154–227.

Makar, R.; Mahadevan, S.; and Ghavamzadeh, M. 2001. Hierarchical multi-agent reinforcement learning. In *Proceedings of the Fifth International Conference on Autonomous Agents*.

Marsella, S.; Adibi, J.; Al-Onaizan, Y.; Kaminka, G.; Muslea, I.; and Tambe, M. 1999. On being a teammage: Experiences acquired in the design of robocup teams. In *Proceedings of the Third International Conference on Autonomous Agents*.

Mataric, M. 1997. Reinforcement learning in the multi-robot domain. *Autonomous Robots* 4(1):73–83.

Riley, P., and Veloso, M. 2001. Coaching a simulated soccer team by opponent model recognition. In *Proceedings of the Fifth International Conference on Autonomous Agents*.

Rohanimanesh, K., and Mahadevan, S. 2003. Learning to take concurrent actions. In *Neural Information Processing Systems (NIPS)*.

Shachter, R. D. 1986. Evaluating influence diagrams. *Operations Research* 34(6):871–882.