When Did You Start Doing That Thing That You Do?
Interactive Activity Recognition and Prompting

Yi Chu and **Young Chol Song** and **Henry Kautz**
Department of Computer Science
University of Rochester
Rochester, NY 14627

Richard Levinson
Attention Control Systems
650 Castro Street, PMB 120-197
Mountain View, CA 94041
www.brainaid.com

Abstract

We present a model of interactive activity recognition and prompting for use in an assistive system for persons with cognitive disabilities. The system can determine the user's state by interpreting sensor data and/or by explicitly querying the user, and can prompt the user to begin or end tasks. The objective of the system is to help the user maintain a daily schedule of activities while minimizing interruptions from questions or prompts. The model is built upon an option-based hierarchical POMDP. Options can be programmed and customized to specify complex routines for prompting or questioning.

Novel aspects of the model include (i) the introduction of adaptive options, which employ a lightweight user model and are able to provide near-optimal performance with little exploration; (ii) a restricted-inquiry dual-control algorithm that can appeal for help from the user when sensor data is ambiguous; and (iii) a combined filtering / most likely-sequence algorithm for activities determining the beginning and ending time points of the user's activities. Experiments show that each of these features contributes to the robustness of the model.

1   Introduction

People with cognitive impairments can often improve their ability to adhere to a regular schedule of activities by using systems that provide timely prompts and reminders (Levinson 1997; Riffel et al. 2005; Bushnik et al. 2007). While today's commercially-available prompting tools are essentially enhanced calendars, researchers have recognized the need for context-aware systems that infer the user's state and provide the right kind of help at the right time (Pollack et al. 2002; Rudary, Singh, and Pollack 2004; Lundell et al. 2007; Modayil et al. 2008; Mihailidis and Boqer 2008). A context-aware prompting system can avoid unnecessary or incorrect prompts and reduce confusion and learned dependency.

This paper presents a model of interactive activity recognition and prompting. The goals of the system are to ensure that the user adheres to a daily schedule by providing prompts to begin or end activities, to create logs of the user's activities, and to minimize interruptions to the user. It differs from previous work in that it includes both prompts and information-seeking actions (inquiries to the user) in its action space, and is able to arbitrate between these two kinds of actions. User responses are used to help identify states and improve the activity recognition model.

The model builds upon a hierarchical POMDP (partially-observed Markov decision process) that runs on an underlying MDP. The hierarchy exploits the problem structure and speeds up learning by reducing the action space at each level. One challenge in the prompting domain is the wide variety of possible prompting methods and the need for temporally-extended prompts. For example, it may be desirable to repeat a prompt every five minutes until the user responds, and/or to increase the level of detail in the prompting instructions. We address this challenge by using temporally extended actions called *options* (Sutton, Precup, and Singh 1999). The system is able to specify complex prompting behavior by directly programming different options, while maintaining a small action space. Furthermore, we introduce *adaptive options* that can rapidly adapt their behavior based on user modeling. The adaptive option implements a light-weight learning process without subjecting the user to long period of policy exploration as in a standard reinforcement learning.

Our second contribution is to propose an effective heuristic solution for the POMDP. Solving the full POMDP would require extensive computation over an enormous state-space (for a model-based approach) and/or extensive training instances (for a reinforcement-learning approach). Our model employs a selective-inquiry based "dual control" approach to dealing with uncertainty. The algorithm assesses the uncertainty in the current estimation of the state, and goes into different control modes based on the uncertainty level. We argue that this approach is well suited to our problem and can be used on-line effectively.

Our third contribution is to combine on-line filtering and most-likely sequence (MLS) inference in order to accurately and efficiently estimate the beginning and end time of activities. The state estimator sends the controller the marginal probability of each activity and the most likely time the activity would have begun if it were the true activity. This information allows the controller to determine when to prompt, while avoiding the state explosion that would oc-

cur if the system attempted to track the cross product of all activities and all starting and ending times.

We evaluate the model by comparing our approach to various alternatives. We run the adaptive option and three other fixed options on three types of simulated users with different behavior patterns, and show that the adaptive option not only adapts to particular user behaviors quickly, but also maintains the best performance across all scenarios. We compare the unified selective-inquiry dual control approach with models that always or never inquire when the state is ambiguous, and that use ordinary filtering rather than filtering/MLS estimation. The results show that the unified dual control approach consistently achieves the most robust performance.

2 The Model

The system starts the day with an initial schedule of *tasks* that the user needs to perform. Following (Pollack et al. 2002; Rudary, Singh, and Pollack 2004), the schedule is constructed and revised as necessary by an interactive constraint-based interval (Smith, Frank, and Jónsson 2000) planning system. Each task has an associated time window over which it can be performed, a minimum and maximum duration, and a target starting time. The *state estimator* inputs data from devices such as IR motion sensors, RFID object touch sensors (Smith et al. 2005), and appliance operation sensors, and outputs a probability distribution over the user's possible activities. The *controller* incorporates information from both the state estimator and schedule, and selects system behaviors. If the world is fully observable (*i.e.*, the state estimate is a point estimate), the control architecture can be modeled as a Markov decision process (MDP); more generally, it is a partially-observed Markov decision process (POMDP), or equivalently, an MDP over belief states.

Because the set of belief states is infinite, an exact solution to a POMDP is in general intractable. Simple greedy heuristics for POMDPs include assuming the most likely state is the true state, or assuming that the value of a belief state is the weighted sum of the underlying MDP states (the Q-MDP heuristic). Such heuristics reduce to arbitrary strategy selection when uncertainty is high. An important feature of our application domain, however, is that the system is able to reduce uncertainty by *asking the user* what he or she is doing. This suggests the use of a *dual control rule* (Cassandra 1998) that queries the user in the face of uncertainty. However, in order to avoid interrupting the user unnecessarily, we propose a *selective-inquiry* rule that only attempts to reduce uncertainty when doing so would cause the controller to select different actions.

2.1 Options in Hierarchical MDPs

Our domain is highly structured because actions (prompts to begin or end activities) are restricted to particular situations. Hierarchical representations can exploit this problem structure. Hierarchy in our model is represented through temporally-extended actions called *options* (Sutton, Precup, and Singh 1999). Each option is defined with its own internal policy, a set of possible initial states, and termination conditions. Each task is associated with a set of options, and each option runs over the state space of its associated task. The task *status* determines which of its options are enabled, that is, which are available to be chosen by the controller for execution. (Note that enabled options might be chosen for execution, but do not *necessarily* execute.)

Task status is determined according to the schedule and the current state. For example, a task T is *ready* when the current time step is within the start window of T and T has not started yet. A *start prompt* option for the task can only be initiated when the task becomes *ready*. The termination of an option or the initiation of a new option always occurs with the change in task status. From inactive, a task may transit to *active* (either *ready* or *underway*), and from active to *completed* or *failed*.[1] An option is executed based on its internal policy; in this domain, its *prompting strategy*. A prompting strategy defines all aspects regarding the generation of a prompt, such as timing, modality, specificity and so on. With a detailed prompting strategy, we are able to specify very complex prompting behavior in a compact and highly customized way.

Because options are defined only over the task space, the hierarchical learning problem is decomposed into a collection of independent learning problems, where each task runs its own MDP. Q-learning over options (Sutton, Precup, and Singh 1999) is done over each MDP individually. At each time step, the controller updates the state, terminates the currently executing option if its termination conditions are met, and then selects an option with highest utility for execution from the set of available options for the active task.

2.2 Adaptive Options

A key aspect of a prompting strategy is its timing. It is of particular interest not only because it is a critical feature for the performance of a prompting behavior, but also because it is not trivial to optimize. An optimal strategy should avoid prompting the user too soon (that is, before the user has a chance to self-initiate the task) as well as too late (that is, risking task failure). Learning to choose among a large set of different options with different timings could require a lengthy training process. Furthermore, bad prompts that result from exploring the policy space can frustrate the user.

To overcome this limitation, we introduce *adaptive options* that rapidly adapt prompting behavior based on a lightweight user model. We consider two kinds of user variables that can affect the timing of a prompt, *initiative* and *responsiveness*. Initiative indicates how soon the agent will initiate a task without any prompt, and responsiveness reflects how long the user will take to respond to a prompt. Thus, if initiative and responsiveness are both short, the system should spend more time waiting for the desired task being self-initiated before issuing a prompt, and vice versa. In this way, we are trading off between two objectives: trying to avoid unnecessary prompts and ensuring the task occurs in time.

---

[1]Our implemented system also handles task interruptions by using a *suspended* task status; we omit the description of interruptions because of lack of space.

Suppose a task T is scheduled to start within the time interval [ES, LS]. Let $F_1(t)$ represent the cumulative probability of the user initiating a task within t steps since ES. Let $F_2(t)$ represent the cumulative probability of the user starting the task within t steps since a prompt. If a prompt is scheduled at time $t_3 \in$ [ES, LS], the probability that the agent initiating T is denoted as $P_1 = F_1(t_3 - ES)$. Similarly, the probability of the agent responding to a prompt in time (before LS) is $P_2 = F_2(LS - t_3)$. The expected reward obtained if we prompt at t_3 is therefore

$$E(R|t_3) = P_1\overline{R}_1 + (1 - P_1)(P_2\overline{R}_1 + (1 - P_2)\overline{R}_2 - c) \quad (1)$$

where $\overline{R}_1$ and $\overline{R}_2$ are the expected cumulative rewards obtained when T has *started* and *failed* respectively, and c is the cost of the prompt. Recall that $\overline{R}_1$ and $\overline{R}_2$ are available, because they are exactly the result of Q-learning, *i.e.*, $\overline{R}_1 = V(status = started)$ and $\overline{R}_2 = V(status = failed)$. The time point that maximizes the equation (1) is the estimated optimal prompt time t_p. Note that different kinds of variations can be added into this computation of $E(R|t_3)$ to reflect specific needs or preferences. For example, delay cost can be included if we wish to emphasize the user's adherence to schedule.

However, a problem arises in learning $F_1(t)$ from user behavior. While $F_1(t)$ is the time until the user self-initiates, in many trials the user will be prompted, and thus not have a chance to self-initiate. We do not want to ignore these trials in estimating $F_1(t)$; they tell us that *if* the prompt had not been issued at t, the self-initiation time would have *exceeded* t. Techniques for estimating cumulative probability distributions from this kind of "right-censored" data has been developed in work on reliability (Crowder et al. 1991). When data is subject to right censoring, $F(t)$ is not learned directly, but instead is modeled by the survivor function $S(t) = 1 - F_1(t)$. We employ the Kaplan-Meier estimate, which works by identifying k distinct time points when the observed event occurs: $t_1, t_2, ..., t_k$. Let n_j be the number of trials when the event is observed to occur at time t_j, and m_j the number of trials that are alive at t_j. Then the Kaplan-Meier estimate of $S(t)$ is given by

$$\hat{S}(t) = \prod_j (1 - \frac{n_j}{m_j}) \quad (2)$$

where $t_j < t$. The empirical estimate of $S(t)$ does not depend on any specific probability model, but it suggests the form of a likely model for the data, namely, a Weibull distribution. With an appropriate choice of parameters, a Weibull can take on the characteristics of many other types of distributions. The simulation results show that an estimated Weibull model helps to learn the pattern of the user behavior even when the actual distribution is Gaussian.

2.3 Uncertainty Reasoning: Dual Control

The user's *CurrentActivity* is determined (in the absence of explicit inquiries) by a Hidden Markov Model (HMM). The states of the HMM are the possible activities associated with tasks, and an "other activity" state. The observations in the HMM are the sensor streams, including object touches as determined by a wearable RFID reader and location as determined by motion sensors. On-line filtering computes a probability distribution over the set of activities. Activity recognition using an HMM and this kind of sensor data can be quite reliable and accurate (Smith et al. 2005; Patterson et al. 2005). In most but not all cases, the output of the HMM is close to a point-estimate. The fact that uncertainty is significant but limited in scope motivates the use of a control mechanism that is computationally simpler than solving a full POMDP.

The basic idea of dual control mechanism is straightforward: when the state uncertainty is small, the state with the most probability mass is the true state; if the state uncertainty is large, the system can choose an action to reduce the uncertainty — in our domain, it can query the user. The *normalized entropy* (Cassandra 1998) of a probability distribution b is computed as $\overline{H}(b) = \frac{H(b)}{H(u)}$, where $H(b)$ is the entropy of the probability distribution b and $H(u)$ is the uniform distribution. Normalized entropy is in the range of $0 \leq \overline{H}(b) \leq 1$ because the maximized entropy is achieved for $H(u)$. We choose a small threshold δ. If $\overline{H}(b) \leq \delta$, the *CurrentActivity* is updated to be the most likely state. Otherwise, the state is considered to be ambiguous.

It is not, however, always appropriate to query the user when the state becomes ambiguous. Such interruptions have an inherent cost to the user. It might be better, for example, to simply wait and see if further observations eliminate the ambiguity. Intuitively, an inquiry action is not needed when no other action than *wait* is available considering all possible states. This critical observation can be easily proved by formalizing the value of actions, $V(b, inquiry)$ and $V(b, wait)$, in terms of the Q functions of the underlying MDP. Based on the above discussion, we propose the selective-inquiry based dual control (SI-DU) algorithm (Alg.1). The key to the successful implementation of the algorithm is to decide the critical point when an inquiry is needed. As talked about in the previous section, our model employs adaptive options to compute the approximately optimal time point for generating a prompt. And here we use a simplified mechanizm by justifying the issue of an inquiry based on this *best* point for taking a system action. It is also possible to take more complex approaches by further weighing the value of an inquiry with the other available actions (note this can be readily added to the algorithm by locally modifying line 28-32 in Alg.1).

However, to make the adaptive options work effectively, it is necessary to record the exact time point when an event occurs, *e.g.*, the task starts. Such timing information is important for both learning the correct user model and computing the optimal prompt time. But in a selective-inquiry based model, the state ambiguities are not resolved immediately and it is very likely that a critical point of *state transition* is already passed when the current state becomes disambiguated again. Our model deals with this problem by combining online filtering with maximum likely-sequence (MLS) estimation.

2.4 Combining Filtering and MLS

In the SI-DU algorithm, the function *Update_Task_Status* updates the task status based on the current state. This also

Algorithm 1 Selective-inquiry based Dual Control (SI-DU)

```
      Input:
 2:   b: the current belief of the world
      δ: thresh-hold on H̄(b)
 4:   T_list: the list of tasks
      Return:
 6:   action: the system action based on b

 8:   At each time step t:
      if Get confirmed state s after an inquiry then
10:      Set_HMM (s)
         H̄(b) ← 0
12:   end if
      if H̄(b) < δ then
14:      s ← argmax_s b(s)
         status(T_list) ←Update_Task_Status(s)
16:      action ← π_{MDP}(s ∪ status(T_list))
      else
18:      {Decide whether to issue an inquiry or wait}
         S^n ← the set of n most likely states based on b
20:      A ← ∅ {Init the set of permissible actions based on
         b}
         for s ∈ S^n do
22:         status(T_list) ←Update_Task_Status(s)
            a ← π_{MDP}(s ∪ status(T_list))
24:         if a ≠ wait then
               A ← A ∪ a
26:         end if
         end for
28:      if A ≠ ∅ then
            action ← inquiry
30:      else
            action ← wait
32:      end if
      end if
```

includes updating information about the actual start or end time of the activity associated with the task. When the state is fully observed, knowledge of *CurrentActivity* is sufficient for determining timing information. However, as we have argued, some activities are not disambiguated immediately.

One solution to this issue would be to track the cross product of activities and starting time of the activity (or equivalently, the cross product of activities and durations), but this would be computationally expensive. A practical alternative is to determine the time point at which an activity begins by reasoning backward from the time at which the activity is strongly believed to hold: that is, when the entropy of the state estimate is below the threshold δ. When a state becomes unambiguous in this manner, we calculate the most likely start time (MLST), most likely end time (MLET), and most likely duration (MLD) of prior activities given the *CurrentActivity*. We define

- $MLST(A, M)$ as the latest timestep t where $a_{t-1} \neq A$ and $a_t = A$ in the most likely sequence ending at M
- $MLET(A, M)$ as the latest timestep t where $a_t = A$ and $a_{t+1} \neq A$ in the most likely sequence ending at M

- $MLD(A, M)$ as the time difference t between $MLET(A, M)$ and $MLST(A, M)$

The most likely sequence ending at activity M is given as, $\operatorname{argmax}_{a_1,...,a_{T-1}} P(a_1, ..., a_{T-1}, o_1, ..., o_T | a_T = M)$. This can be efficiently computed using the Viterbi algorithm. We also introduce a method to feed back information gathered from the inquiry to the state estimator. When a particular state is confirmed by the user, the system "fixes" that state (*Set_HMM* in Alg.1) to hold with probability 1.

3 Experiments

We conducted two sets of experiments with a simulated user and environment. In the first set of experiments, we are trying to demonstrate the effectiveness of the adaptive option by comparing its learning process with different fixed options. In the second set of experiments, we compare the result of five models with variations to illustrate how the unified dual control approach outperforms other models.

Experiment I. In this experiment, we simulated different kinds of user behaviors with different levels of initiative and responsiveness. Focusing on the start option, the simulated user can initiate a task at any time point within [ES, LS] by sampling from a distribution modeling initiative. After a prompt, the user behavior changes, and a start point is sampled instead from the distribution of responsiveness. *type I* user has high initiative and responsiveness. The Weibull modeling the initiative falls almost entirely into the earlier part of the window. Responsiveness is also modeled with Weibull and set to a high value: the user responds within 5 steps of a prompt 90% of the time. The *type II* user is as highly responsive as *type I*, yet with a much lower initiative, where *initiative* is modeled as a normal distribution with the mean as the midpoint of the start window and a relatively large variance. Compared with *type II*, the *type III* user has the same initiative model but a lower responsiveness. The user responds to a prompt within 5 steps only 70% of the time.

We compared four kinds of strategies: no prompt at all, prompt at the earliest time (ES), prompt at the latest time (LS-5) and the adaptive strategy. In the learning process, the system penalizes each prompt with a small cost (-2), the failure of a task with a large cost (-10), and rewarded the successful completion of a task with a reward (+10). To make a relatively consistent setting for all of the experiments, we assume that once a task is started, it is always completed successfully. We ran experiments on all three types of users for 10 times. In each experiment, Q-learning is updated for 60 iterations. We observe how the averaged utility of each kind of option changes when the user exhibits different patterns of behavior. Note that the utility of different options reflect the effectiveness of different strategies.

Obviously, the best strategy for type I user is to prompt as late as possible or not to prompt at all. The average results from 10 runs show that the adaptive option adapts to the type I user as well as the other two best fixed options (no prompt and latest prompt strategies). The results for type II & III users are displayed in Fig. 1. In both scenarios, adaptive option stands out as the best strategy. A task has a scheduled
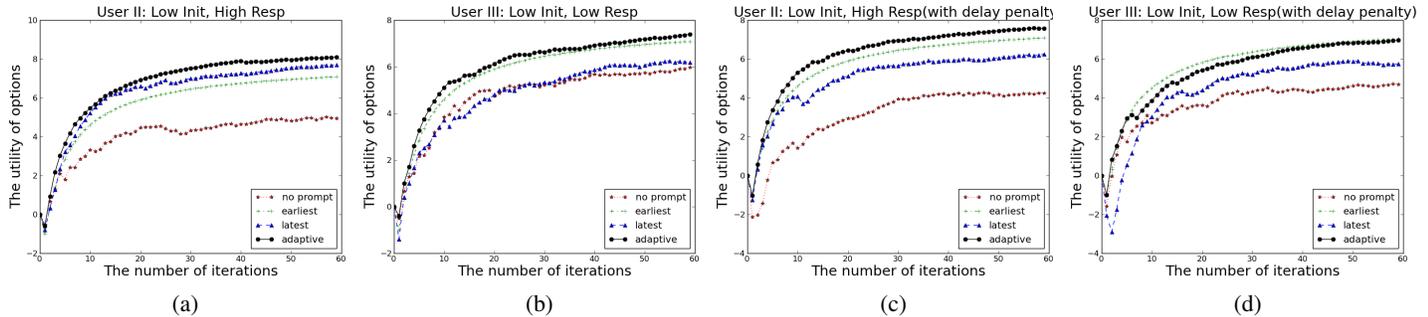
Figure 1: The utility of options for the type II and type III user.

or preferred execution time, t_p, and the system is designed to improve the user compliance with the schedule. To reflect this preference, the system is penalized every time a task is started at a point later than t_p. The adaptive option explicitly takes the delay cost into the computation of the expected reward. The results in (Fig. 1c & 1d) show that the adaptive option not only adapts to particular user behavior, but also to the user preferences. It should be noted although we are only talking about start options here, the same approach applies to other options (*e.g.*, option for resuming a task) or the second prompt in the same option.

Experiment II. In this experiment, we are simulating a partially observable environment in order to test dual control. Considering a HMM with three states: breakfast (BF), take-medicine(TM) and no activity (NO). The set of observations (RFID object touches) we have for BF is {'cupboard', 'cup', 'spoon', 'cerealbox'}, and for TM is {'cupboard', 'cup', 'spoon', 'medicine'}. We create a situation where only objects common to both BF and TM are observed. This can occur in the real world in situations where the tags for cerealbox or medicine fails, or when the RFID reader misses readings. In addition, we introduce a special state called NO, representing a state when the user randomly touches objects with certain probability (10%) without initiating a particular task. We simulate a simple scenario where the user starts a task and then ends it. As mentioned ealier, the task has a start window as well as the scheduled start (t_p) and end time (t_e). Two kinds of prompts are available: start prompt (prompt the user to start a task) and a stop prompt (prompt the user to end a task if it is *under way* and $t > t_e$). For this experiment, we are comparing five different models: Model I is our unified model; the other alternative models are:

- Model II (never inquiry): When $\overline{H} > \delta$, the system system just waits instead of issuing an inquiry.
- Model III (always inquiry): The system always issues an inquiry immediately when $\overline{H} > \delta$.
- Model IV (no filtering combined with MLS): The system did not use backward MLS to estimate the start/end of the task.
- Model V (no adaptive option): Instead of adaptive option, the system learns over a set of fixed options with various prompt times.

Table 1 lists the summarized results after 100 iterations for each model in terms of the system actions, user behavior (execution of schedule), and inference. These three classes of measures correspond to the system's objectives. In the experiments, the uncertainty rate is measured by the percent of the time steps when the state is ambiguous ($\overline{H}(b) > 0.4$). When the system makes no explicit actions (inquiry) to resolve uncertainty (model II), the uncertainty rate is around 61%. The cells in gray indicate the performance of the corresponding model is *poor* enough compared with other models. It is clear from this table that Model I performs consistently robust across all measures. It is interesting that model V (where different fixed options are used instead of the adaptive option) misses many prompts and failed more often with its randomness in exploration. In terms of learning, experiments also show that even after 100 iterations, Q learning with fixed options (15 different strategies) is still not converging (an adaptive one takes less than 20 iterations). For other models with the adaptive option, Model II is not learning the correct strategy. Model IV takes longer to converge and learns a strategy earlier than optimal timing.

Model	I	II	III	IV	V
System Action					
# of Inquires per run	1.2	0	4	1.3	1.7
# of prompts per run	0.8	0.7	0.81	0.9	0.7
Prompt error rate (%)	0	15	1	2	5
Prompt miss rate (%)	2	27	2	0	14
Execution of Schedule					
Failure rate (%)	0	3	1	0	3
Ave. start delay (step)	2.9	6.4	2.7	4.4	6.5
Ave. finish delay (step)	3.0	6.3	2.4	3.0	3.9
Inference					
Start infer. failure (%)	0	51	0	2	4
End infer. failure (%)	0	51	0	2	4
Ave. diff (start) (step)	0.2	1.5	0.72	5.0	1
Ave. diff (end) (step)	0.2	1	0.18	1.1	1.3

Table 1: Performance of different models in the face of uncertainty after 100 iterations. Gray cells indicate lowest performance.

4   Related Work

Our work is inspired by the growing interest in the development of intelligent prompting systems, and in particular by the systems PEAT (Levinson 1997), Autominder (Pollack et al. 2002), and COACH (Boger et al. 2005). PEAT was the first system to combine prompting with automated planning and rescheduling. Autominder introduced the vision of a unified, context-aware prompting agent, and COACH introduced the use of a POMDP model for prompting an individual through a task. (Rudary, Singh, and Pollack 2004) modeled prompting as a reinforcement learning problem where the set of available actions is controlled by the user's daily schedule, and is thus a primary inspiration for our model. However, the RL takes long time to converge and the uer is thus suffering from inconsistent and erroneous system behavior. (Modayil et al. 2008) proposes a high-level architecture for context-aware prompting that is compatible with our model. (Vurgun, Philipose, and Pavel 2007) showed that a context-aware medication prompting system that used a rule-based deterministic control strategy could improve medication adherence. This system, along with PEAT and Autominder, does not involve advanced prompting technology and high-level uncertain state reasoning. POMDPs have been used to provide a rich framework for planning under uncertainty in assistive systems (COACH (Boger et al. 2005)). Various approximations for POMDPs have been developed that can be solved off-line (Poupart 2005; Varakantham, Maheswaran, and Tambe 2005). However, these algorithms scale poorly when the state space becomes very large, while our heuristic approach may handle problems with a large group of activities with each activity devided into very fine-grained sub-states or sub-steps.

The high accuracy of HMMs for activity recognition from object touch RFID data was demonstrated by (Smith et al. 2005; Patterson et al. 2005) and other researchers. Hui and Boutilier applies a Bayesian approach to inferring a variety of variables that reflect a user's personality and attitudes, *e.g.*, neediness, frustration, independence, *etc*, and use this model to determine the kind of assistance to offer to the user. Finally, Fern et al. applied the Q-MDP approximation to solving a novel POMDP model of an intelligent assistant, but did not consider inquiry actions for resolving uncertainty.

5   Conclusions and Future Work

We presented a POMDP model of interactive activity recognition and prompting that is made efficient and practical through the use of dual control to handle uncertainty, adaptive options to reduce training time, and combined filtering/most likely sequence estimation to estimate activity timing information. We presented simulation results that showed that each of these features improved the performance of the model.

An important issue in the activity recognition and prompting domain not discussed in this paper is the need to handle task interruptions and resumptions. A common problem in task performance by persons with cognitive disabilities is neglecting to resume a task that has been interrupted. We

are extending the model described in this paper to allow it to distinguish activities that have completed from activities that are suspended, and to support options that prompt to resume an activity.

We have implemented our model in an Android phone application, and are in the process of designing and carrying out an evaluation of the system with approximately 10 patients at the Palo Alto Veterans Administration outpatient clinic who have brain injury, PTSD, pre-dementia, or other cognitive impairment. The implementation is based on a modified version of the commercial prompting system PEAT (Levinson 1997), and employs sensor data from motion sensors, contact sensors, and wearable RFID readers. The initial set of tasks to be supported include waking up and going to bed, meals, taking medicine, and performing therapy homework. Prompts and queries to the user are delivered by audio and graphics on the phone. Surveys delivered on the cell phone and observational studies will be used to evaluate the accuracy and effectiveness of the system. We will compare these real-world results with the results of our simulation studies, and use them to create better models for use in building future prototypes.

6   Acknowledgements

References

Boger, J.; Poupart, P.; Hoey, J.; Boutilier, C.; Fernie, G.; and Mihailidis, A. 2005. A decision-theoretic approach to task assistance for persons with dementia. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK*, 1293–1299.

Bushnik, T.; Dowds, M.; Fisch, J.; and Levinson, R. 2007. Current evidence-base for the efficacy of electronic aids for memory and organization. In *American Congress on Rehabilitation Medicine (ACRM)*.

Cassandra, A. R. 1998. *Exact and approximate algorithms for partially observable markov decision processes*. Ph.D. Dissertation, Brown University, Providence, RI, USA.

Crowder, M. J.; Kimber, A. C.; Smith, R. L.; and Sweeting, T. 1991. *Statistical analysis of reliability data*. Chapman & Hall.

Fern, A.; Natarajan, S.; Judah, K.; and Tadepalli, P. 2007. A decision-theoretic model of assistance. In Veloso, M. M., ed., *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India*, 1879–1884.

Hui, B., and Boutilier, C. 2006. Who's asking for help?: a bayesian approach to intelligent assistance. In *IUI 06: Proceedings of the 11th international conference on Intelligent user interfaces*, 186–193. New York, NY, USA: ACM.

Levinson, R. 1997. The planning and execution assistant and trainer (peat). *Journal of Head Trauma Rehabilitation* 12(2):85–91.

Lundell, J.; Hayes, T.; Vurgun, S.; Ozertem, U.; Kimel, J.; Kaye, J.; Guilak, F.; and Pavel, M. 2007. Continuous activity monitoring and intelligent contextual prompting to improve medication adherence. In *29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*.

Mihailidis, A., and Boqer, J. 2008. The coach prompting system to assist older adults with dementia through handwashing: An efficacy study. *BMC Geriatrics* 8(1):28.

Modayil, J.; Levinson, R.; Harman, C.; Halper, D.; and Kautz, H. 2008. Integrating sensing and cueing for more effective activity reminders. In *AAAI Fall 2008 Symposium on AI in Eldercare: New Solutions to Old Problems.*

Patterson, D. J.; Fox, D.; Kautz, H.; and Philipose, M. 2005. Fine-grained activity recognition by aggregating abstract object usage. In *ISWC '05: Proceedings of the Ninth IEEE International Symposium on Wearable Computers*, 44–51. Washington, DC, USA: IEEE Computer Society.

Pollack, M. E.; Mccarthy, C. E.; Tsamardinos, I.; Ramakrishnan, S.; Brown, L.; Carrion, S.; Colbry, D.; Orosz, C.; and Peintner, B. 2002. Autominder: A planning, monitoring, and reminding assistive agent.

Poupart, P. 2005. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov decision processes.* Ph.D. Dissertation, Univeristy of Toronto, Toronto, Cananda.

Riffel, L.; Wehmeyer, M.; Turnbull, A.; Lattimore, J.; Davies, D.; and Stock, S. 2005. Promoting independent performance of transition-related tasks using a palmtop pc-based self-directed visual and auditory prompting system. *Journal of Special Education* 20(2).

Rudary, M. R.; Singh, S. P.; and Pollack, M. E. 2004. Adaptive cognitive orthotics: combining reinforcement learning and constraint-based temporal reasoning. In *ICML 2004: Machine Learning, Proceedings of the Twenty-first International Conference, Banff, Alberta, Canada*, volume 69. ACM.

Smith, J. R.; Fishkin, K. P.; Jiang, B.; Mamishev, A.; Philipose, M.; Rea, A. D.; Roy, S.; and Sundara-Rajan, K. 2005. Rfid-based techniques for human-activity detection. *Communications of the ACM* 48(9).

Smith, D. E.; Frank, J.; and Jónsson, A. K. 2000. Bridging the gap between planning and scheduling. *Knowl. Eng. Rev.* 15:47–83.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning. *Artif. Intell.* 112(1-2):181–211.

Varakantham, P.; Maheswaran, R. T.; and Tambe, M. 2005. Exploiting belief bounds: practical pomdps for personal assistant agents. In *4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), July 25-29, 2005, Utrecht, The Netherlands*, 978–985.

Vurgun, S.; Philipose, M.; and Pavel, M. 2007. A statistical reasoning system for medication prompting. In *UbiComp 2007: Ubiquitous Computing, 9th International Conference, Innsbruck, Austria*, 1–18. Springer.