Journal of Ambient Intelligence and Smart Environments 0 (0) 1
IOS Press

Interactive Activity Recognition and Prompting to Assist People with Cognitive Disabilities

Yi Chu [a], Young Chol Song [a], Richard Levinson [b] and Henry Kautz [a]
[a] *Department of Computer Science, University of Rochester, Rochester, NY 14627, US*
[b] *Attention Control Systems, 650 Castro Street, PMB 120-197, Mountain View, CA 94041, US, www.brainaid.com*

Abstract. This paper presents 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, resume, 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.

The paper proposes a heuristic approach to solving the POMDP based on a dual control algorithm using selective-inquiry that can appeal for help from the user explicitly when the sensor data is ambiguous. The dual control algorithm is working effectively in the unified control model which features the adaptive option and robust state estimation. Simulation results show that the unified dual control model achieves the best performance and efficiency comparing with various alternatives. To further demonstrate the system's performance, lab experiments have been carried out with volunteer actors performing a series of carefully designed scenarios with different kinds of interruption cases. The results show that the system is able to successfully guide the agent through the sample schedule by delivering correct prompts while efficiently dealing with ambiguous situations.

Keywords: assistive system, context-aware prompting, activity recognition, MDP, POMDP, ambient intelligence

1. Introduction

The fast growing population of the ageing society will result in a dramtic increase in the number of people diagnosed with cognitive disabilities (such as Alzheimer's disease or other forms of dementia). People with cognitive disabilities suffer from memory loss and executive function impairment [19,12,33] that prevent them from organizing, managing or carrying out everyday tasks independently. Commonly, they are experiencing failures such as failing to initiate a task, forgetting an unfinished task after interruptions, performing the tasks incorrectly and so on. To compensate for deficits in cognitive function, human assistance is needed to provide regular prompts that help the patient through the activities of daily living. The constant dependence and pressure on the caregivers have a negative impact on both the patient and caregiver that could lead to diminished quality of life, increased level of anxiety, poor self-esteem, and social isolation [2].

To support independent living and reduce the cost of heath care, researchers have developed various technologies and computing systems that can automate the prompting behavior and alleviate the burden on caregivers. This could be electronic devices that provide timely prompts and reminders to support schedule adherence and time management [13,24,3]. 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 [22,25,15,19,17]. For instance, prompting to start a task that the person has already started could be confusing. Similarly, a prompt for taking medication

is not effective if the person is on the phone but works if he is engaging in the leisure activity, e.g., watching TV. A context-aware prompting system can sense aspects of the user context and adapt its prompting behavior accordingly. Further, studies show that frequent interventions can overwhelm an individual's cognitive resource capacity, thus reducing task effort and producing negative effects on performance [35,43]. A context-aware prompting system can avoid unnecessary and incorrect prompts in order to minimize the cost of interruptions.

While prompting systems are a valuable tool for assistive technology, prompt strategies need to consider the independence aspect of the goal. As pointed out in [37,38], the independence of older adults may be compromised by well meaning nursing facility staff giving excessive prompts. So they present a restorative care approach to help foster independence by giving the mininum assistance possible through a strategy called *the system of least prompts (SLP)* [37]. SLP is a systematic method of fading prompts that is widely used in the setting of special education to help individuals with metal retartation and other developmental disbilities [44,39,40]. Across literature, the importance of fading prompts is universally recognized to help promote prompt-free or independent performance [36,44,41]. The adverse effects of the uncautionary use of prompts may include prompt dependency, contrary control and consequence confusion [41,42]. While in these studies human assistance is typically playing the primary role of controlling the prompt procedure, it is equally important for a computer system to learn to restrict the use of prompts when it is designed to entirely or partially replace the human effort. As with the training of human staff, a context-aware prompting system can also be trained so that it delivers prompts only when needed through the learning of the user's ability in initiating a correct behavior and his responsiveness to prompt.

To perform well an intelligent prompting system must be able to infer the state of the world, reason about the costs of varying system actions, handle uncertainties about the environment and the user, and to adapt to the user behavior pattern. This depends on the coordinated function of various components working together. This paper introduces a unified framework that integrates sensing, scheduling, planning, prompting and the user in a complete cycle. In [19], Modayil talks about the benefits of integrating sensing into an existing planning and cueing device PEAT [13] and proposes a high-level architecture for context-aware prompting that is compatible with our model. To address the challenges involved in developing such an integrated system, a model of interactive activity recognition and prompting is built which is based on partially observable Markov decision processes (POMDPs). POMDPs use a decision-theoretic approach to optimizing the course of system actions over a user-specified utility function that is compatible with the system's objectives despite incomplete state information. The goals of the system are to ensure that the user adheres to a daily schedule by providing prompts to begin, resume, 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.

To get a quick overview of how the system works, consider the scenario based on the simple schedule shown in table 2. Suppose the day starts with the task breakfast (BF). From the collected data of behavior patterns, the system learns that the user usually has no trouble remember having breakfast on his own. So the system spent some time waiting for the user to initiate the task before generating a prompt. After detecting that the user has started preparing breakfast, the system terminates the prompting routines pertaining to starting BF. Meanwhile, the user's activity is interrupted by an incoming phone call after he just puts bread on the toaster. After the call, the user forgets about the toaster or bread and goes to watch TV. The system identifies the suspension of BF and generates a prompt for continuing the activity with breakfast. The user goes back to preparing breakfast, eats it, and finishes cleaning up. However, when the scheduled end time of BF has passed, the system still couldn't decide on the current status of the task. To resolve ambiguity, the system asks the user of his situation. On receiving the user reply, the system terminates the prompting routine for stopping BF to avoid confusion. Then it waits for a while and at a proper time reminds the user of taking his medicine (TM). The user quickly responds to the prompt and the schedule is completed.

In the rest of the paper, we first introduce the challenges and contributions in developing the model. This is followed by a detailed description of the model, focusing on the central controller. Then evaluation results are presented that are based on both simulation data and lab experiments to demonstrate the model's

efficiency and robustness. Finally the paper talks about the related work and makes a conclusion.

2. Our contributions

The model builds upon a hierarchical POMDP (partially-observed Markov decision process) that runs on an underlying MDP [28]. 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. To address this challenge temporally extended actions are used that are called *options* [29]. The system is able to specify complex prompting behavior by directly programming different options, while maintaining a small action space. Furthermore, *adaptive options* are introduced, which can rapidly adapt their behavior to different users 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. The paper argued that this approach is well suited to our problem and can be used effectively on-line.

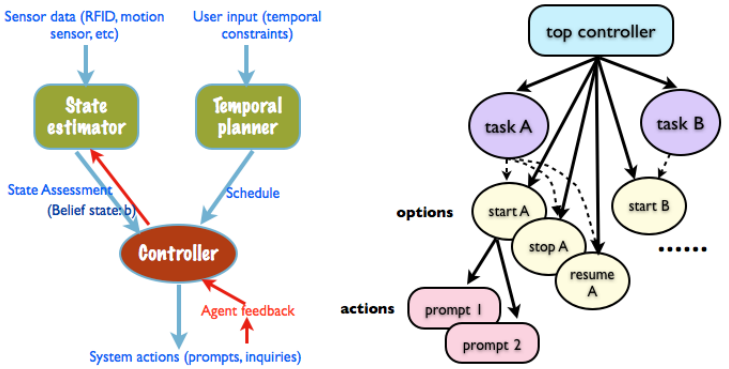
Our third contribution is to combine on-line filtering and most-likely sequence (MLS) inference in order to accurately and efficiently retrieve the time point of backward events. The state estimator sends the controller the marginal probability of each activity and the most likely time the activity would have begun or ended if it were the true activity. This information allows the controller to decide task status and determine when to prompt. One of the common problems that the people with cognitive disabilities have is forgetting about resuming a suspended task. Our model is able to determine the time point when a task is interrupted, distinguish it from completed status, and send out a prompt for resuming the task at an appropriate time.

The model is evaluated with both simulation based results and human subject experiments. The adaptive option and three fixed options are run 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. The paper compares the unified selective-inquiry dual control approach with alternative models in a simulated environment. The results show that the unified dual control approach consistently achieves the most robust performance. Finally, experiments are carried out with human subjects performing a series of carefully designed scenarios involving different cases of interruptions and demonstrate the system's ability to generate proper prompts.

3. The Model

The system starts the day with an initial schedule of *tasks* that the user needs to perform. Following [22,25], the schedule is constructed and revised as necessary by an interactive constraint-based interval [26] 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 time. The *state estimator* inputs data from devices such as IR motion sensors, RFID object touch sensors [27], and appliance operation sensors, and outputs a probability distribution over the possible states of the world. The *controller* incorporates information from both the state estimator and schedule, decides per task *status*, and selects appropriate system behaviors (fig. 1(a)). User feedback is also fed into the controller and used to help identify the state and improve the activity classification model. 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 [4]). Such



(a) System Overview (b) Hierarchical Controller based on Options

Fig. 1. Illustration of the Model Structure

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* [4], which queries the user in the face of uncertainty. However, in order to avoid interrupting the user unnecessarily, a novel *selective-inquiry* rule is proposed, which only attempts to reduce uncertainty when doing so would cause the controller to select different actions.

3.1. Temporal Planning

The planner is built on the constraint-based interval (CBI) [26] formalism which works by solving a set of temporal constraints over the time windows of a set of tasks. These temporal constraints are typically entered by a caregiver using a simple graphical interface at the start of the day. Such constraints are defined over variables denoting the start (S) and end point (E) of each task. These constraints represent the temporal relations between different event points, *e.g.*, the constraint for "task A starts after task B" is represented as $B_S - A_E > 0$, and "task A lasts less than 30 minutes" as $A_E - A_s < 30 * 60$, where each time unit is one second. With an absolute reference time point, the absolute temporal constraints regarding the execution of a task is formalized in the same way. If a task has preconditions, the planner will try to resolve the open preconditions and add in new tasks if necessary. The idea of using planning over temporal constraints to support a cognitive aid was first introduced by the PEAT ([13]) and Autominder ([22]). Our system is more general, however, in that it can handle user preferences as well as hard constraints.

The planning algorithm returns a consistent set of constraints determining the partial ordering between different tasks. After solving the constraints using Bellman-Ford algorithm, each event point is bounded by an earliest time and a latest time, e.g., a start point is bounded within [ES, LS], where ES is the earliest possible start time and LS is the latest possible start time. To obtain an exact schedule that reflects user preferences (indicated by the caregiver or the resident), slack variables are introduced to represent preference constraints. Suppose the user indicates the preferred time for starting a task T as PS, then this time preference is encoded with two constraints: $T_S + \zeta_1 \geq PS$ and $T_S - \zeta_2 \leq PS$, where $\zeta_1, \zeta_2 \geq 0$. The preferred duration (PD) is programmed in a very much similar way with the other two slack variables η_1 and η_2. These constraints are added into the original set of constraints. By solving a linear optimization problem over all the constraints with the objective of minimizing $\zeta_1 + \zeta_2 + \eta_1 + \eta_2$, the exact time schedule is obtained for executing the task T that best reflects the user's preferences.

3.2. Decision Making in Hierarchical MDPs

Our domain is highly structured because actions (*e.g.*, prompts to begin or end activities) are restricted to particular situations. Hierarchical representations can exploit this problem structure and accelerate learning by partially hardcoding the policy space. Hierarchy in our model is represented through temporally-extended actions called *options* [29], where each op-

tion decides a sequence of primitive *actions* to be followed until it is terminated.

3.2.1. Options and Tasks

An option is defined with its own internal policy, a set of possible initial states (determined by its initiation conditions), 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. From ready, a task may transit to *underway* or *failed*, and from *underway* to *completed* or *failed*. A task is said to be active when it is *ready* or *underway*. The completed transitions of task status are illustrated in the Fig. 2. The termination of an option or the initiation of a new option always occurs with the change in task status and the execution of schedule, which is declared explicitly as termination or initiation conditions of the option. For instance, the initiation conditions of a *stop* option dictate that the option is only enabled when the task is *underway* and the current moment is greater than the scheduled end time. And a *start* option can only be initiated when the task becomes *ready*. Our implemented system also handles task interruptions by identifying a *suspended* task status and enabling the *resume* option.

An option is executed based on its internal policy, which implements the option in terms of primitive *actions* that are at the lowest level of the calling hierarchy; in this domain, its *prompting strategy*. A prompting strategy defines all aspects regarding the generation of a prompt action, such as timing, modality, specificity and so on. With a detailed prompting strategy, the model is 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 [29] is done over each MDP individually. Options with different prompting strategies are considered as distinct ones, so the utility of an option reflects the effectiveness of its associated strategy.

At each time step, the controller updates schedule and the status of all the scheduled tasks, checking the termination conditions of the currently executing option if there is any, and selects one (if no other option is running) with highest utility for execution from the set of available options of the active tasks. It is possible that there might be more than one active tasks going on, and the task with higher priority is chosen over the lower priority one. Noted that an executing option can be forcibly terminated by initiating a new option of a higher priority task. Once an option is selected for execution, it decides a primitive *action* to be generated based on its prompting strategy. This hierarchical decision making structure is illustrated in the Fig. 1(b).

3.2.2. Adaptive Options

A key aspect of a prompting strategy is its timing. 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). However, learning to choose among a large set of different fixed 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, *adaptive options* are introduced that rapidly adapt prompting behavior based on a lightweight user model. Two kinds of user variables are considered 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, the model is trading off between two objectives: trying to avoid unnecessary prompts and ensuring the task occurs in time.

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 prompt is generated at t_3 is therefore

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

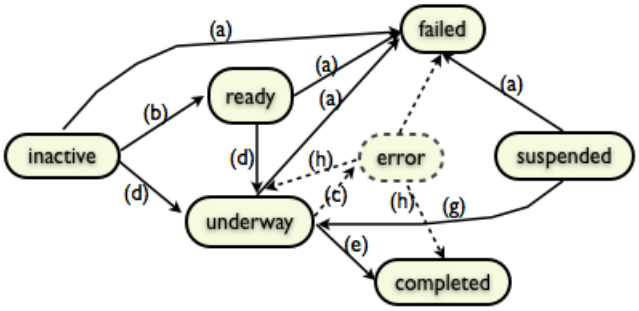


Fig. 2. The simplified view of the status transitions for a task. The cause of transition is displayed beside the transition arrow. *Error* status (framed with dashed line) is currently not fully implemented in the model.

(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.*, $\overline{R}_1 = V(status = \textit{started})$ and $\overline{R}_2 = V(status = \textit{failed})$. 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 the model needs 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. These trials are not to be ignored 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 [5]. When data is subject to right censoring, $F(t)$ is not learned directly, but instead is modeled by the survivor function $S(t) = 1 - F_1(t)$. The Kaplan-Meier is employed 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.

3.3. Uncertainty Handling

So far the paper has talked about the decision-making process on a MDP basis, but the problem the system is dealing with is fraught with uncertainties and incomplete information. In the model, 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 [27,21] 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 [27,21]. 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.

3.3.1. Dual Control

The basic idea of dual control mechanism is straightforward: when the state uncertainty is small, the state

Input:
b: the current belief of the world
δ: thresh-hold on $\overline{H}(b)$
T_list: the list of tasks
Return:
action: the system action based on b

At each time step t:
if Get confirmed state s after an inquiry **then**
  setHMM (s)
  $\overline{H}(b) \leftarrow 0$
end if
if $\overline{H}(b) < \delta$ **then**
  $s \leftarrow \text{argmax}_s\, b(s)$
  $status(T\_list) \leftarrow$**UpdateTaskStatus**(s)
  action $\leftarrow \pi_{MDP}(s \cup status(T\_list))$
else
  {*Decide whether to issue an inquiry or wait*}
  $S^n \leftarrow$ the set of n most likely states based on b
  $A \leftarrow$ {*Init the set of permissible actions based on b*}
  for $s \in S^n$ **do**
    $status(T\_list) \leftarrow$**UpdateTaskStatus**(s)
    $a \leftarrow \pi_{MDP}(s \cup status(T\_list))$
    if $a \neq wait$ **then**
      $A \leftarrow A \cup a$
    end if
  end for
  if A contains different actions **then**
    action $\leftarrow inquiry$
  else
    action $\leftarrow$ any $a \in A$
  end if
end if

Fig. 3. Selective-inquiry based Dual Control algorithm (DC-SI)

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* [4] of a probability distribution b is computed as $\overline{H}(b) = H(b)/H(u)$, 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 $H(u)$.A small threshold δ is chosen. If $\overline{H}(b) \leq \delta$, the current world state 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. In the algorithm shown in Fig. 3, A heuristic mechanism is employed to decide whether to issue an inquiry: an inquiry is needed when different possible states result in the selection of different actions. It is possible to integrate more quantitative approaches to further weigh the value of asking a question against the values of other available actions (by locally modifying the line 28 in the DC-SI algorithm (Fig. 3). Further analysis on the value of asking a question is given in the next section to examine how this works.

3.3.2. Value of Inquiry

Suppose H_1 and H_2 are two hypotheses about the state of world at the current moment t based on the state belief b. H_1 is true with probability $b(s_1)$ when the world is in the state s_1, and similarly H_2 is true with probability $b(s_2)$. a_1 and a_2 are two best actions to address each state. Suppose a_1 is non-trivial (other than 'wait'), the expected value of a_1 can be represented as the weighted sum of the underlying states:

$$\begin{aligned} V(b, a_1, t) &= b(s_1) * V(H_1, a_1, t) + b(s_2) * V(H_2, a_1, t) \\ &\approx b(s_1) * Q(s_1, a_1, t) \\ &\quad + b(s_2) * (E + Q(s_2,' wait', t)) \end{aligned} \tag{3}$$

where E is the constant denoting the penalty for erroneous action, *i.e.*, delivering an incorrect prompt. Here two heuristics are employed: first, a myopic strategy is used to approximate $V(H_1, a_1, t)$ based on the Q functions of the underlying MDP model; secondly, $V(H_2, a_1, t)$ is approximated by the sum of two parts: the cost of error action a_1 in state s_2 and the value of being idle (waiting instead of taking action a_2) in s_2.

The value of an inquiry action is comprised of three parts: the cost of inquiry action (C), which might be a little more expensive than a normal prompt, the value of taking the correct action based on user response $V(correct)$, and the value of waiting $V(wait)$ when no reply is obtained. Note that the user responsiveness to an inquiry can be learned the same way as to a prompt (in Sec. 3.2.2). Suppose the user responds to a query within t time steps with probability $F(t)$, and the expected delay is DD based on $F(t)$. Let TT be the

time-out condition of an inquiry, the expected value of *inquiry* is thus estimated as,

$$\begin{aligned} V(b, inquiry, t) &= F(TT) * V(correct) \\ &+ (1 - F(TT)) * V(wait) \\ &+ C \end{aligned} \quad (4)$$

$$\begin{aligned} V(correct) &\approx b(s_1) * Q(s_1, a_1, t + DD) \\ &+ b(s_2) * Q(s_2, a_2, t + DD) \end{aligned} \quad (5)$$

$$\begin{aligned} V(wait) &\approx b(s_1) * Q(s_1,' wait', t) \\ &+ b(s_2) * Q(s_2,' wait', t) \end{aligned} \quad (6)$$

The dynamics of online reasoning about belief and action under time-pressured context depends on the uncertainty level, the task urgency (time to deadline), and task importance (task reward). With the same belief state, the advantage of asking a question over the other actions (reflected by the difference between Eq. 3 and 4) diminishes as the task is increasing in urgency or importance, making the decision on the best action less likely to be *inquiry*. Based on the values of different actions, the choice of action (line 28 in the DC-SI algorithm) is determined by the formula

$$action \leftarrow \underset{a}{\operatorname{argmax}} V(b, a, t), a \in A \cup wait \cup inquiry$$

3.3.3. State Estimation: Retrieval of Event Points

The key to the successful implementation of the dual control algorithm is to determine the critical time point when an inquiry is needed. Adaptive options estimate the approximately optimal time point for generating a prompt, which can be considered as a decision point for an inquiry. However, to make the adaptive option work effectively, it is necessary to record the exact time point when an event occurs, (*e.g.*, the task starts), which is needed for both learning the correct user model and computing the prompt time. In addition, the controller needs to identify the start, end, suspension or resumption times of a task in order to update its status and decide options. However, in a selective-inquiry based model, state ambiguities are not necessarily resolved immediately. It is therefore possible that a critical *state transition* has already passed when the current state is finally disambiguated. In such cases, the algorithm needs to retrieve the time points of the missed events.

The activity model is built based on a HMM model where the state space is the set of all the possible activities. When the state is fully observed, knowledge of *CurrentActivity*, obtained from on-line filtering, is sufficient for determining timing information. However, as 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, the most likely start time (MLST), most likely end time (MLET), and most likely duration (MLD) of prior activities are calculated given the *CurrentActivity* M. They are defined as

- MLST(A, M) as the latest time step 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 time step 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 MLS(M) ending at activity M is given by,

$$MLS(M) = \underset{a_1,...,a_{T-1}}{\operatorname{argmax}} P(a_1, ..., a_{T-1}, o_1, ..., o_T | a_T = M). \quad (7)$$

This can be efficiently computed using the Viterbi algorithm. A method is introduced 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 (*setHMM()* in the DC-SI algorithm (Fig. 3) to hold with probability 1.

Thus, by examining the most likely state sequence, the system successfully recovers the time points when task status changes. This information is then used to further update the user model, which in turn is used to estimate the optimal timing of a prompt action in an adaptive option.

3.3.4. Interruption Model

When a task is executed without interruptions or not interleaved with the other activities, its completion is clearly identified as the most likely end time. However, in the real world, when the user performs a task,

he may suspend the task before finishing and go on to another task. The presence of interruptions and interleaving activities poses challenge for recognizing the completion of a task. The same difficulty holds for the differentiation between the start and resumption of a task.

In order to correctly trace the progress of the scheduled tasks in the case of interruptions, the per activity state is divided into sub states for each activity, including an ending state that signifies the completion of an activity. In the extended interruption model, the state space of HMM comprises the sub-states of all the activities. Initially, the transition probabilities in the model favor a non-interruption model, where the sub-activities occur in sequence, and the transition to the next activity is highly likely only when the current activity is at its final state. However, if the MLS determines an activity has been interrupted, the model takes note of the interrupted sub-state and assign a higher transition probability to resuming at that sub-state. The transition probabilities of our interruption model can be described as having so-called logical transitions [20], where the transitions to the successor states (resuming to the start or a certain sub-state of an activity) depends on the current activity status (inactive, active, or suspended).

The model records information that keeps track of the interruption and resumption time point of each task: the most recent suspend time (MRST), the most recent suspend state (MRSS), the most recent resume time (MRRT), and the most likely init time (MLIT). MRSS is the very last state just before a task switches out of its internal sub-states. If MRSS equals the ending state, the task is identified as completed, otherwise suspended. MRST and MRRT are determined in the same way as MLET and MLST in a model without interruptions, as described in the previous section. MLIT is the point when the activity is first initiated. If the activity is executed uninterruptedly to its end, MLIT = MRRT = MRST. Otherwise, MRRT gives the time point when the task is resumed after its most recent suspension.

4. Simulation Results

Two sets of experiments are conducted with a simulated user and environment. The purpose of the first set of experiments is to demonstrate the effectiveness of the adaptive option by comparing its learning process with different fixed options. In the second set of experiments, the results of five models with variations are compared to illustrate how the unified dual control approach outperforms other models.

4.1. Experiment I: Adaptive Option

In this experiment, different kinds of user behaviors are simulated 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. The *type I* user has high initiative and responsiveness. The Weibull modeling the *initiative* falls almost entirely into the earlier part of the window. The *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. The user is more erratic in terms of selecting the timing for executing a task and can probabilistically forget about doing the task entirely. 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.

Four kinds of strategies are compared: 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, it is assumed that once a task is started, it is always completed successfully. Experiments are run on all three types of users for 10 times. In each experiment, Q-learning is updated for 60 iterations. The results show 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

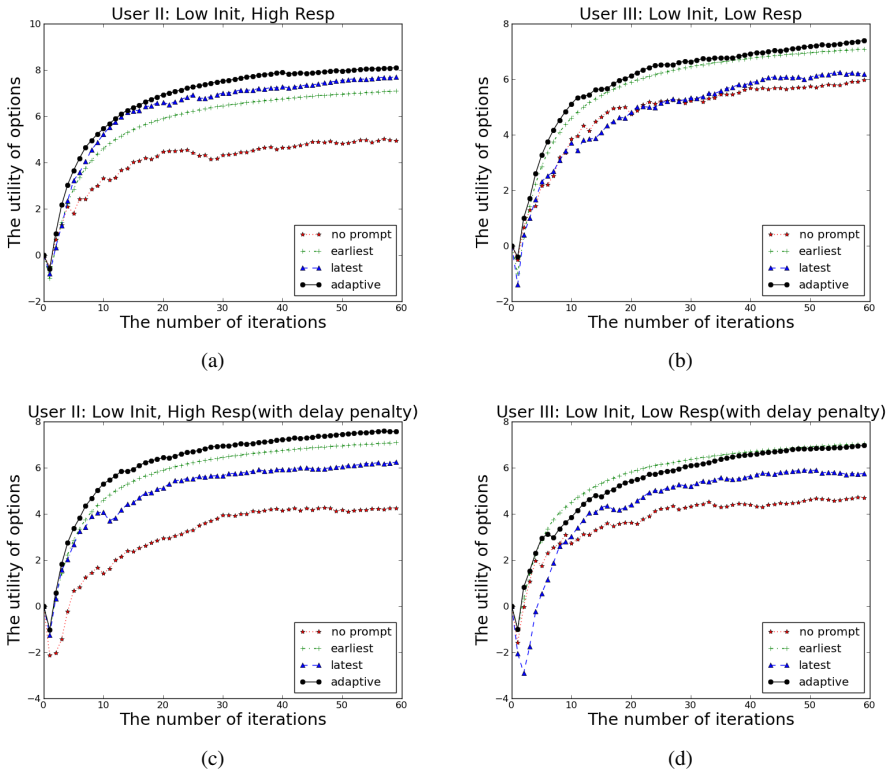


Fig. 4. The utility of options for the type II and type III user.

in Fig. 4. In both scenarios, the adaptive option stands out as the best strategy.

A task has a scheduled 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 the Fig. 4(c) and 4(d) show that the adaptive option not only adapts to particular user behavior, but also to the user preferences. It should be noted although the paper is 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.

4.2. Experiment II: Unified Control Model

In this experiment, partially observable environment is simulated 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) for BF is {cupboard, cup, spoon, cereal-box}, and for TM is {cupboard, cup, spoon, medicine}. A situation is created where only objects common to both BF and TM are observed. This can occur in the real world in situations where the tags for 'cereal-box' or 'medicine' fails, or when the RFID reader misses readings. In addition, a special state is introduced called NO, representing a state when the user randomly touches objects with certain probability (10%) without initiating a particular task. A simple scenario is simulated where the user starts a task and then ends it. As mentioned earlier, 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, five different models are compared: Model I is our unified control 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	**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. discrepancy (start time) (step)	0.2	1.5	0.72	5.0	1
Ave. discrepancy (end time) (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 (smaller numbers, better performance).

– Model IV (no filtering combined with MLS): The system did not use MLS to retrieve the time point of backward events.
– 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. Model II failed in all three measurements. Model III achieved sound performance but at the cost of too many interactive questions. Model IV did poorly in the exact inference. 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, despite the fact that the adaptive option is well trained in 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 with earlier than optimal timing.

5. Experiments with Human Subjects

To demonstrate how the system works in terms of generating prompts and asking questions, two volunteer actors (students), who have no prior knowledge of how the system works, are asked to walk through a series of scenarios in our lab. Each volunteer wore RFID bracelets on both hands and performed two tasks according to a simple schedule (Table 2). The participants were asked to respond to all reminding prompts, *i.e.*, to do exactly as the system instructed. The test scenarios are designed to include interruption cases where a task is suspended before completion and then resumed later. In the first scenario, the participants followed the scheduled tasks sequentially without any interruptions, *i.e.*, they start with the task of breakfast, finish it and then take medicine. In the second and third scenarios, the participants stopped the breakfast halfway to initiate other tasks, either taking medicine or watching TV, and then went back to finish breakfast (after the system prompt). The general criteria for testing the success of the system was that the system could behave properly, *i.e.*, generate appropriate prompts in the right situation and ask questions when needed. In

Task	Start window	scheduled start	scheduled end
BF	[0, 30]	15	115
TM	[120, 150]	135	175

Table 2
A simple test schedule

general, the system was able to successfully guide the agent through the schedule by instructing them to start, finish or resume a task. The detailed results are described in the tables.

Tables 3-8 summarize the transcript of three test scenarios by two participants. This includes the user behavior, inferred state, system action and its appropriateness. To track the progress of breakfast and medicine, each activity is divided into sub stages: BF_B (preparing breakfast), BF_M (eating), BF_E (cleaning up), TM_B (getting medicine), TM_M (taking medicine), TM_E (putting away medicine). In the first scenario, the system delivered prompts to start the task and stop the task when it was not finished as scheduled. One incorrect prompt occurred when the system didn't recognize the ending state of breakfast due to a period of consecutive missing readings from the sensors. At the beginning, the system waited for a while and then prompted to start breakfast in the middle (step=15) of the start window based on the schedule. When the system detected that the user had started breakfast, it updated task status and terminated the start option so that no more prompts would be generated for starting breakfast. Noted that during the very first runs, when no knowledge of the user's behavior pattern is learned, the system usually sends out a prompt in the middle of the given time window within which the system can afford to wait for the user to initiate the task on his own. In the case of a resume prompt, the system estimated the approximate time left for the suspended task to be completed based on the elapsed period of time spent over it so far and the prior knowledge of task duration. This information, combined with the schedule, is used to decide the time window for a resume prompt. Alternatively, a stop prompt is usually generated at the scheduled ending time of the task if it is not finished by then, The contents of prompts were written that reflect the intents of the different kinds of prompt. During the run time, the correct prompt content is decided by the system and sent out using the text-to-speech APIs supported by Mac OS X Lion Speech engine. Questions regarding the user activity is sent out in a very much similar way. And the user can input feedbacks by selecting from a list of choices through the GUI on the computer.

In the second scenario, the agent stopped eating breakfast and went to take medicine. Since having medicine is considered a higher priority task, the system waited until the agent finished the medicine and then prompted him to resume the breakfast task. In the third scenario, the agent went to watching TV in the middle of breakfast. The system dutifully prompted him to stop TV and return to breakfast after some time. Throughout all the experiments, the system issued 20 prompts in total, one of which is erroneous (sensor error). There are in all 7 inquiries in the presence of a total of 172 ambiguous time steps, and all the inquiries are reasonably justified based on the real situation. The result demonstrated the system's ability in recognizing activities, generating proper prompts and handling interruptions and resumptions. It also demonstrated the model's effectiveness and efficiency in dealing with ambiguous situations.

6. Related Work

Our work is inspired by the growing interest in the development of intelligent prompting systems, and in particular by the systems PEAT [13], Autominder [22], and COACH [1]. PEAT was the first system to combine prompting with automated planning and rescheduling. However, PEAT is primarily developed as a scheduling aid and its ability of tracking the progress of schedule relies on the user's self-reports. Autominder introduced the vision of a unified, context-aware prompting agent. However, both PEAT and Autominder have not demonstrated how automatic sensing can be integrated into the system to provide more effective prompts, and neither do they develop mechanisms to handle uncertainties in state estimation.

Vurgun [32] showed that a context-aware medication prompting system that used a rule-based deterministic control strategy could improve medication adherence. However, the system is sending out prompts based on a set of pre-defined rules, and makes no effort in exploring the most fruitful prompting strategy. There has been research on studying how to learn effective, adaptive prompting strategies that do the most good. Rudary [25] learned the timing of prompt us-

Step	**Real Situation**	**Inferred State**	**System action**		**Notes**
15	BF: ready; TM: not ready;	NO	prompt to start BF	$\surd$	
30	Start preparing BF;	NO			
86	Finish cleaning up breakfast;	BF_B			Sensor error; no readings while eating using a spoon
95	BF: completed; TM: not ready;	NO	prompt to resume BF	$\times$	State estimate error (see above): BF was falsely estimated to be suspended at state BF_B
135	BF: completed; TM: ready;	NO	prompt to start TM	$\surd$	
143	Start getting medicine;	NO			
170	Finish putting away medicine;	TM_E			
173	BF: completed; TM: completed;	NO, BF_B	issue inquiry	$\surd$	Disambiguate: if BF_B is true, prompt to stop BF.
175	(same as above)	NO			Get reply
Participant A: Number of ambiguous steps: 21 (12%)					

Table 3
Transcript of Scenario I (Participant A): no interruption

Step	**Real Situation**	**Inferred State**	**System action**		**Notes**
15	BF: ready, TM: not ready;	NO	prompt to start BF	$\surd$	
19	Start preparing BF;	NO			
114	Finish cleaning up BF;	MD_B			State estimated incorrectly, but recovered later
135	BF: completed, TM: ready	NO	prompt to start TM	$\surd$	
138	Start getting medicine;	NO			
169	BF: completed, TM: started	TM_E, BF_B	issue inquiry	$\surd$	Disambiguate: if BF_B is true, prompt to stop BF.
173	(same as above)	TM_E			Get the reply
176	(same as above)	TM_E	prompt to stop TM	$\surd$	
181	Finish putting away medicine;	TM_E			
Participant B: Number of ambiguous steps: 27 (15%)					

Table 4
Transcript of Scenario I (Participant B): no interruption

Step	Real Situation	Inferred State	System action		Notes
15	BF: ready, TM: not ready	NO	prompt to start BF	√	
18	Start preparing BF;	NO			
51	Suspend eating BF;	BF_B, BF_M			No inquiry was needed.
56	Start getting medicine;	NO			
81	Finish putting away medicine;	TM_E			
83	BF: suspended, TM: completed	NO, TM_E	issue inquiry	√	Disambiguate: if NO is true, prompt BF to resume.
87	(same as above)	NO			Get reply
88	(same as above)	NO	prompt to resume BF	√	
90	Resume eating BF;	NO			
116	BF: resumed, TM: completed	BF_E	prompt to stop BF	√	
135	Finish cleaning up BF;	BF_E			
137	BF: completed, TM: completed	BF_E, NO	issue inquiry	√	Disambiguate: if BF_E is true, prompt to stop BF.
140	(same as above)	NO			Get reply
Participant A: Number of ambiguous steps: 31 (22%)					

Table 5
Transcript of Scenario II (Participant A): interruptions by higher-priority task

Step	Real Situation	Inferred State	System action		Notes
15	BF: ready, TM: not ready	NO	prompt to start BF	√	
18	Start preparing BF;	NO			
66	Suspend eating BF;	BF_E			
103	Finish putting away medicine;				
106	BF: suspended, TM: completed	TM_E, NO	issue inquiry	√	Disambiguate: if NO is true, prompt to resume BF
110	(same as above)	NO	prompt to resume BF	√	Get reply
112	Resume eating BF;	NO			
131	BF: resumed, TM: completed	BF_M	prompt to stop BF	√	
159	Finish clearing up BF;	MD_E, MD_B			No inquiry is needed (ambiguation resolved itself quickly).
Participant B: Number of ambiguous steps: 38 (24%)					

Table 6
Transcript of Scenario II (participant B): interruptions by higher-priority task

Step	Real Situation	Inferred State	System action		Notes
15	BF: ready, TM: not ready	NO	prompt to start BF	√	
19	Start preparing BF;	NO			
49	Suspend eating BF;	BF_B, BF_E			No inquiry needed
56	Start watching TV;	NO			
63	(same as above)	NO, TV	prompt to resume BF	√	
70	Resume eating BF;	NO			
85	Suspend eating BF again;	BF_M			
93	(same as above)	NO	prompt to resume BF	√	
97	Resume eating BF;	NO			
112	Finish cleaning up BF;	BF_B			
116	(same as above)	NO, BF_B	issue inquiry	√	Disambiguate: if BF_B is true, prompt to stop.
119	(same as above)	NO			Get reply
135	BF: completed, TM: ready	NO	prompt to start TM	√	
139	Start getting medicine;	NO			
164	Finish putting away medicine;	TM_B			
167	BF: completed, TM: completed	NO, TM_M			No inquiry needed (ambiguation resolved itself quickly).
Participant A: Number of ambiguous steps: 21 (13%)					

Table 7
Transcript of Scenario III (Participant A): interruptions by lower priority task

ing reinforcement learning where the set of available actions is controlled by the user's daily schedule. The learned prompting strategy adapts to different users and is thus a primary inspiration for our model. However, its reinforcement learning algorithm takes long time to converge, and during this period the user would suffer from inconsistent and erroneous system behavior. [10,17] are studying how to effectively automate the generation of prompting actions using a decision-theoretic approach. The system is tailoring the prompting behavior, in terms of both the timing and level of specificity, to specific individuals. Das [6] is trying to learn the timing of prompts by directly mining from a large set of sensor data. However, they are not differentiating the prompting strategies for different users. Neither do they consider the cost of prompts.

A successful prompting system should be able to recognize the user state with high accuracy, develop effective interaction strategies that can vary and adapt to different user behaviors and contexts, and make reasonable decisions in the face of all kinds of certainties inherent in such a problem. So far, the most sophisticated kind of this system is COACH [1], which introduced the use of a POMDP model for prompting an individual through a task, i.e., hand-washing, by giving step by step instructions. POMDPs provide a powerful framework for dealing with uncertainties and utilities in a theorectically well-justified manner, which have been applied successfully to a set of prompting systems that focus on one particular aspect of daily support, e.g., art therapy (ePAD [9]), Stroke Rehabilitation (iSTRETCH [11]), or a specific kitchen task such as making tea (SNAP [8]). But solving a full POMDP is computationally intensive. Various approximations for POMDPs have been developed that can be solved offline [23,31]. However, these algorithms scale poorly when the state space becomes very large, making it difficult or even intractable to be deployed in the real setting. Our heuristic approach may efficiently handle problems with a large group of activities with each activity divided into very fine-grained sub-states or substeps. Fern [7] applied the Q-MDP approximation to solving a novel POMDP model of an intelligent assistant, but did not consider inquiry actions for resolving uncertainty.

Step	Real Situation	Inferred State	System action		Notes
10	Start preparing BF;	NO			
57	Suspend eating BF;	BF_M			
59	Start watching TV	BF_M			
73	Watching TV; BF: suspended, TM: not ready;	TV	prompt to resume BF	√	
80	Resume eating BF;	NO			
116	BF: resumed, TM: not ready	BF_E	prompt to stop BF	√	
137	BF: completed, TM: ready	NO	prompt to start TM	√	
142	Start getting medicine; BF: completed, TM: started	NO			
162	BF: completed, TM:started	NO, BF_B	issue inquiry	√	If NO is true, TM ended; if BF_B is true, prompt to start TM.
176	(same as above)	TM_M	prompt to stop TM	√	Didn't get the reply; state is self-resolved.
184	Finish putting away medicine; BF: completed, TM: completed	TM_E			
Participant B: Number of ambiguous steps: 24 (13%)					

Table 8
Transcript of Scenario III (participant B): interruptions by lower-priority task

Our work is also related to monitoring systems that achieve tracking and activity recognition with a pervasive sensor infrastructure. One of the essential services required for a context-aware prompting system is an accurate tracking and activity recognition system that reveals critical information about the user's context. There have been an impressive amount of work in using various sensors including GPS, cameras, RFID, and infrared or ultrasound badges to track people's activities. For example, Liao [14] has shown how to extract a person's activities and significant places from traces of GPS data. Recently, the high accuracy of HMMs for activity recognition from object touch RFID data was demonstrated by [27,21] and other researchers. Researchers at MIT [30] studies the potential of using a large number of simple, low cost sensors for accurate activity recognition in the home. Furthermore, they are exploring the simultaneous tracking and activity recognition (STAR) [34] problem for automatic health monitoring in the home environment. In [18], Mileo is proposing a logic-based context model for monitoring the user's quality of life, level of acitivity and health state through the aggregation and the interpretation of different kinds of information from heterogeneous sources (such as light, position, movement, localization, load cells, etc.) The system is also designed to identify risky situations, *i.e.*, fall detection, and provide prompts for prevention through declarative policies. In [16], Moran is trying to understand and predict the undesirable effects, such as increases in stress, of ubiquitous monitoring technology on the user by developing a preliminary model consisting of a series of factors believed to influence user behavior.

7. Conclusions and Future Work

This paper presents a hierarchical POMDP model of interactive activity recognition and prompting. The model is expressive enough to support the design of a prompting system that handles uncertainty, queries to the user, and multiple prompting options. To keep the size of the problem at a contained and practical level, the dual control is used to handle uncertainty, adaptive options to reduce training time and the combined filtering/most likely sequence estimation to infer the timing of past events. Simulation results are presented that showed that the unified model combining all of these features outperforms alternatives. A common problem in task performance by persons with cognitive disabili-

ties is failing to resume a task that has been interrupted. Our model distinguishes activities that have completed from activities that are suspended, and supports options that prompt to resume an activity. The paper also presents a series of lab experiments with human subjects to demonstrate the system's ability to generate correct prompts, ask questions for identifying ambiguous situations, and handle interruptions.

There are several possible extensions for this work. For one thing, it is interesting to see how the model can be personalized to meet the individual user's preferences. It would be nice if the user can have choices over the generation of system's actions bases on their likes or dislikes, e.g., disabling all the prompts or inquiries, or using visual prompts instead of audio ones. This can be easily achieved by giving the user authority to change the parameters or alter the routine of the option policy directly though the user interface. As have been talked about in section 3.2.1, an option policy can be used to specify various aspects of a prompt, including its timing, modality, specificity and so on. Although the focus of this work is not on experimenting with various prompting methods, we believe this is an important function that could be readily integrated into the existing model and improve the system's usability and acceptability. Another extension is to extend the activity model with a set of execution constraints, including preconditions, invariant conditions, and termination conditions so that the model can detect the incorrect execution of a task and help the user recover from *error* status. Finally, the knowledge from the user's schedule could be used to improve the activity classification. One possible approach is to reset the possibility distribution over the plausible activities at the next time step based on the information given by the schedule which tells us what activities are likely to occur.

The model has been implemented 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 [13], 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. These real-world results will be compared with the results of our simulation studies, and use them to create better models for use in building future prototypes.

Acknowledgements

This work is sponsored by DARPA award W31P4Q-09-C-0476, the Office of the Secretary of Defense (OSD) award W81XWH-08-C-0740, and NIH award 1R21HD062898-01.

References

[1] Jennifer Boger, Pascal Poupart, Jesse Hoey, Craig Boutilier, Geoff Fernie, and Alex Mihailidis. 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*, pages 1293–1299, 2005.

[2] Alistair Burns and Peter Rabins. Carer burden in dementia. *International Journal of Geriatirc Psychiatry*, 15:S9–S13, 2000.

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

[4] Anthony Rocco Cassandra. *Exact and approximate algorithms for partially observable markov decision processes*. PhD thesis, Brown University, Providence, RI, USA, 1998.

[5] Martin J. Crowder, A. C. Kimber, R. L. Smith, and T.J. Sweeting. *Statistical analysis of reliability data*. Chapman & Hall, 1991.

[6] Barnan Das, Diane J. Cook, Maureen Schmitter-Edgecombe, and Adriana M. Seelye. Puck: an automated prompting system for smart environments: toward achieving automated prompting-challenges involved. *Personal and Ubiquitous Computing Theme Issue on Sensor-driven Computing and Applications for Ambient Intelligence*, (779), 2011.

[7] Alan Fern, Sriraam Natarajan, Kshitij Judah, and Prasad Tadepalli. A decision-theoretic model of assistance. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India*, pages 1879–1884, 2007.

[8] Jesse Hoey, Thomas Plötz, Dan Jackson, Andrew Monk, Cuong Pham, and Patrick Olivier. Rapid specification and automated generation of prompting systems to assist people with dementia. *Pervasive Mob. Comput.*, 7:299–318, 2011.

[9] Jesse Hoey, Krists Zutis, Valerie Leuty, and Alex Mihailidis. A tool to promote prolonged engagement in art therapy: design and development from arts therapist requirements. In *Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility*, ASSETS '10, pages 211–218, 2010.

[10] Jennifer Boger Ibbme, Jennifer Boger, and Craig Boutilier. A decision-theoretic approach to task assistance for persons with dementia. In *In IJCAI*, pages 1293–1299, 2005.

[11] J.; Kan, P.; Hoey and A. Mihailidis. Stroke rehabilitation using haptics and a pomdp controller. In *AAAI Fall Symposium on Caring Machines: AI in Eldercare*, 2008.

[12] Richard Levinson. A computer model of prefrontal cortex function. *Annals of the New York Academy of Sciences*, 769:381–388, 1995.

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

[14] Lin Liao, Dieter Fox, and Henry Kautz. Extracting places and activities from gps traces using hierarchical conditional random fields. *International Journal of Robotics Research*, 26, 2007.

[15] J. Lundell, T.L. Hayes, S. Vurgun, U. Ozertem, J. Kimel, J. Kaye, F. Guilak, and M. Pavel. 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*, 2007.

[16] Stuart Moran and Kenchi Nakata. Ubiquitous monitoring and user behavior: a preliminary model. *Journal of Ambient Intelligence and Smart Environments*, 2(1):67-80, 2010.

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

[18] Alessandra Mileo, Davide Merico and Roberto Bisiani. Support for context-aware monitoring in home healthcare. *Journal of Ambient Intelligence and Smart Environments*, 2(1):49-66, 2010.

[19] Joseph Modayil, Rich Levinson, Craig Harman, David Halper, and Henry Kautz. Integrating sensing and cueing for more effective activity reminders. In *AAAI Fall 2008 Symposium on AI in Eldercare: New Solutions to Old Problems*, 2008.

[20] Sriraam Natarajan, Hung H. Bui, Prasad Tadepalli, Kristian Kersting, and Weng-Keen Wong. Logical hierarchical hidden markov models for modeling user activities. In *Proceedings of the 18th international conference on Inductive Logic Programming*, ILP '08, pages 192–209, Berlin, Heidelberg, 2008.

[21] Donald J. Patterson, Dieter Fox, Henry Kautz, and Matthai Philipose. Fine-grained activity recognition by aggregating abstract object usage. In *ISWC '05: Proceedings of the Ninth IEEE International Symposium on Wearable Computers*, pages 44–51, Washington, DC, USA, 2005. IEEE Computer Society.

[22] Martha E. Pollack, Colleen E. Mccarthy, Ioannis Tsamardinos, Sailesh Ramakrishnan, Laura Brown, Steve Carrion, Dirk Colbry, Cheryl Orosz, and Bart Peintner. Autominder: A planning, monitoring, and reminding assistive agent, 2002.

[23] Pascal Poupart. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov decision processes*. PhD thesis, Univeristy of Toronto, Toronto, Cananda, 2005.

[24] L. Riffel, M. Wehmeyer, A. Turnbull, J. Lattimore, D. Davies, and S. Stock. 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), 2005.

[25] Matthew R. Rudary, Satinder P. Singh, and Martha E. Pollack. 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, 2004.

[26] David E. Smith, Jeremy Frank, and Ari K. Jónsson. Bridging the gap between planning and scheduling. *Knowl. Eng. Rev.*, 15:47–83, March 2000.

[27] Joshua R. Smith, Kenneth P. Fishkin, Bing Jiang, Alexander Mamishev, Matthai Philipose, Adam D. Rea, Sumit Roy, and Kishore Sundara-Rajan. Rfid-based techniques for human-activity detection. *Communications of the ACM*, 48(9), 2005.

[28] Richard S.Sutton and Andrew G.Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[29] Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning. *Artif. Intell.*, 112(1-2):181–211, 1999.

[30] Emmanuel Tapia, Stephen Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors pervasive computing. In *In Pervasive*, 3001:158–175, 2004.

[31] Pradeep Varakantham, Rajiv T. Maheswaran, and Milind Tambe. 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*, pages 978–985, 2005.

[32] Sengul Vurgun, Matthai Philipose, and Misha Pavel. A statistical reasoning system for medication prompting. In *UbiComp 2007: Ubiquitous Computing, 9th International Conference, Innsbruck, Austria*, pages 1–18. Springer, 2007.

[33] Joseph Wherton and Andrew Monk. Technological opportunities for supporting people with dementia who are living at home. *International Journal of Human-Computer Studies*, 66:571–586, 2008.

[34] Daniel Wilson and Chris Atkeson. Simultaneous tracking & activity recognition (star) using many anonymous, binary sensors. In *In The Third International Conference on Pervasive Computing*, pages 62–79. Springer-Verlag, 2005.

[35] Chak Fu Lam, D. Scott DeRue, Elizabeth P. Karam and John R. Hollenbeck. The Impact of Feedback Frequency on Learning and Task. Performance: Challenging the "More Is Better" Assumption *Organizational Behavior and Human Decision Processes*, 116:217–228, 2011.

[36] MaryAnn Demchak. Response prompting and fading methods: a review. *American Journal on Mental Retardation*, 94(6):603–615, 1990.

[37] Kimberly K. Engelman, Deborah E. Altus, Michael C. Mosier and R. Mark Mathews. Brief training to promote the use of less intrusive prompts by nursing assistants in a dementia care unit. *Journal of Applied Behavior Analysis*, 36(1):129–132, 2003.

[38] Kimberly K. Engelman, R. Mark Mathews and Deborah E. Altus. Restoring dressing independence in persons with Alzheimer's disease: a pilot study. *American Journal of Alzheimer's Disease and Other Dementias*, 17(1): 37–43, 2002.

[39] Myrna E Libby, Julie S Weiss, Stacie Bancroft and Willian H Ahearn. A Comparison of Most-to-Least and Least-to-Most Prompting on the Acquisition of Solitary Play Skills. *Behavior Analysis in Practice*, 1(1):37-43, 2008 Spring.

[40] Elizabeth A. West and Felix Billingsley. Improving the system of least prompts: a comparison of procedural variations. *Education and Training in Developmental Disabilities*, 40(2):131–144, 2005.

[41] Jane Korsten and Terry Foss. Prompting: a cautionary tale. Kansas City: EMC Communications, Inc., 2011.

[42] Steven F. Maier and Martin E. Seligman. Learned helplessness: theory and evidence. *Journal of Experimental Psychology: General*, 105(1):3–46, 1976.

[43] Avraham N. Kluger and Angelo DeNisi. The effects of feedback interventions on performance: a historical review, a meta-analysis, and a preliminary feedback intervention theory. *Psy-*

chological Bulletin, 119(2):254–284, 1996.

[44] Patricia M. Doyle. System of least prompts: a literature review of procedural parameters. *Journal of the Association for Persons with Severe Handicaps*, 32:107–110, 1998.