A Decision-Theoretic Model of Assistance

Alan Fern and **Sriraam Natarajan** and **Kshitij Judah** and **Prasad Tadepalli**
School of EECS, Oregon State University
Corvallis, OR 97331 USA
{afern,natarasr,judahk,tadepall}@eecs.oregonstate.edu

Abstract

There is a growing interest in intelligent assistants for a variety of applications from organizing tasks for knowledge workers to helping people with dementia. In this paper, we present and evaluate a decision-theoretic framework that captures the general notion of assistance. The objective is to observe a goal-directed agent and to select assistive actions in order to minimize the overall cost. We model the problem as an assistant POMDP where the hidden state corresponds to the agent's unobserved goals. This formulation allows us to exploit domain models for both estimating the agent's goals and selecting assistive action. In addition, the formulation naturally handles uncertainty, varying action costs, and customization to specific agents via learning. We argue that in many domains myopic heuristics will be adequate for selecting actions in the assistant POMDP and present two such heuristics. We evaluate our approach in two domains where human subjects perform tasks in game-like computer environments. The results show that the assistant substantially reduces user effort with only a modest computational effort.

1 Introduction

The development of intelligent computer assistants has tremendous impact potential in many domains. A variety of AI techniques have been used for this purpose in domains such as assistive technologies for the disabled [Boger *et al.*, 2005] and desktop work management [CALO, 2003]. However, most of this work has been fine-tuned to the particular application domains. In this paper, we describe and evaluate a more comprehensive framework for intelligent assistants.

We consider a model where the assistant observes a goal-oriented agent and must select assistive actions in order to best help the agent achieve its goals. To perform well the assistant must be able to accurately and quickly infer the goals of the agent and reason about the utility of various assistive actions toward achieving the goals. In real applications, this requires that the assistant be able to handle uncertainty about the environment and agent, to reason about varying action costs, to handle unforeseen situations, and to adapt to the agent over time. Here we consider a decision-theoretic model, based on partially observable Markov decision processes (POMDPs), which naturally handles these features, providing a formal basis for designing intelligent assistants.

The first contribution of this work is to formulate the problem of selecting assistive actions as an assistant POMDP, which jointly models the application environment and the agent's hidden goals. A key feature of this approach is that it explicitly reasons about the environment and agent, which provides the potential flexibility for assisting in ways unforeseen by the developer. However, solving for such policies is typically intractable and we must rely on approximations. A second contribution is to suggest an approximate solution approach that we argue is well suited to the assistant POMDP in many application domains. The approach is based on explicit goal estimation and myopic heuristics for online action selection. For goal estimation, we propose a model-based bootstrapping mechanism that is important for the usability of an assistant early in its lifetime. For action selection, we propose two myopic heuristics, one based on solving a set of derived assistant MDPs, and another based on the simulation technique of policy rollout. A third contribution is to evaluate our framework in two novel game-like computer environments using 12 human subjects. The results indicate that the proposed assistant framework is able to significantly decrease user effort and that myopic heuristics perform as well as the more computationally intensive method of sparse sampling.

The remainder of this paper is organized as follows. In the next section, we introduce our formal problem setup, followed by a definition of the assistant POMDP. Next, we present our approximate solution technique based on goal estimation and online action selection. Finally we give an empirical evaluation of the approach in two domains and conclude with a discussion of related and future work.

2 Problem Setup

We refer to the entity that we are attempting to assist as the *agent*. We model the agent's environment as a Markov decision process (MDP) described by the tuple $\langle W, A, A', T, C, I \rangle$, where W is a finite set of world states, A is a finite set of agent actions, A' is a finite set of assistant actions, and $T(w, a, w')$ is a transition distributions that represents the probability of transitioning to state w' given that action $a \in A \cup A'$ is taken in state w. We will sometimes use $T(w, a)$ to denote a random variable distributed as $T(w, a, \cdot)$. We assume that the assistant action set always contains the action **noop** which leaves the state unchanged. The component C is an action-cost function that maps $W \times (A \cup A')$ to real-numbers, and I is an initial state distribution over W.

We consider an episodic setting where at the beginning of

each episode the agent begins in some state drawn from I and selects a goal from a finite set of possible goals G. The goal set, for example, might contain all possible dishes that the agent might prepare. If left unassisted the agent will execute actions from A until it arrives at a goal state upon which the episode ends. When the assistant is present it is able to observe the changing state and the agent's actions, but is unable to observe the agent's goal. At any point along the agent's state trajectory the assistant is allowed to execute a sequence of one or more actions from A' ending in **noop**, after which the agent may again perform an action. The episode ends when either an agent or assistant action leads to a goal state. The cost of an episode is equal to the sum of the costs of the actions executed by the agent and assistant during the episode. Note that the available actions for the agent and assistant need not be the same and may have varying costs. Our objective is to minimize the expected total cost of an episode.

Formally, we will model the agent as an unknown stochastic policy $\pi(a|w, g)$ that gives the probability of selecting action $a \in A$ given that the agent has goal g and is in state w. The assistant is a history-dependent stochastic policy $\pi'(a|w, t)$ that gives the probability of selecting action $a \in A'$ given world state w and the state-action trajectory t observed starting from the beginning of the trajectory. It is critical that the assistant policy depend on t, since the prior states and actions serve as a source of evidence about the agent's goal, which is critical to selecting good assistive actions. Given an initial state w, an agent policy π, and assistant policy π' we let $C(w, g, \pi, \pi')$ denote the expected cost of episodes that begin at state w with goal g and evolve according to the following process: 1) execute assistant actions according to π' until **noop** is selected, 2) execute an agent action according to π, 3) if g is achieved then terminate, else go to step 1.

In this work, we assume that we have at our disposal the environment MDP and the set of possible goals G. Our objective is to select an assistant policy π' that minimizes the expected cost given by $E[C(I, G_0, \pi, \pi')]$, where G_0 is an unknown distribution over agent goals and π is the unknown agent policy. For simplicity we have assumed a fully observable environment and episodic setting, however, these choices are not fundamental to our framework.

3 The Assistant POMDP

POMDPs provide a decision-theoretic framework for decision making in partially observable stochastic environments. A POMDP is defined by a tuple $\langle S, A, T, C, I, O, \mu \rangle$, where S is a finite set of states, A is a finite set of actions, $T(s, a, s')$ is a transition distribution, C is an action cost function, I is an initial state distribution, O is a finite set of observations, and $\mu(o|s)$ is a distribution over observations $o \in O$ given the current state s. A POMDP policy assigns a distribution over actions given the sequence of preceding observations. It is often useful to view a POMDP as an MDP over an infinite set of belief states, where a belief state is simply a distribution over S. In this case, a POMDP policy can be viewed as a mapping from belief states to actions. Actions can serve to both decrease the uncertainty about the state via the observations they produce and/or make direct progress toward goals.

We will use a POMDP model to address the two main challenges in selecting assistive actions. The first challenge is to infer the agent's goals, which is critical to provide good assistance. We will capture goal uncertainty by including the agent's goal as a hidden component of the POMDP state. In effect, the belief state will correspond to a distribution over possible agent goals. The second challenge is that even if we know the agent's goals, we must reason about the uncertain environment, agent policy, and action costs in order to select the best course of action. Our POMDP will capture this information in the transition function and cost model, providing a decision-theoretic basis for such reasoning.

Given an environment MDP $\langle W, A, A', T, C, I \rangle$, a goal distribution G_0, and an agent policy π we now define the corresponding *assistant POMDP*.

The **state space** is $W \times G$ so that each state is a pair (w, g) of a world state and agent goal. The **initial state distribution** I' assigns the state (w, g) probability $I(w)G_0(g)$, which models the process of selecting an initial state and goal for the agent at the beginning of each episode. The **action set** is equal to the assistant actions A', reflecting that assistant POMDP will be used to select actions.

The **transition function** T' assigns zero probability to any transition that changes the goal component of the state, i.e., the assistant cannot change the agent's goal. Otherwise, for any action a except for **noop**, the state transitions from (w, g) to (w', g) with probability $T(w, a, w')$. For the **noop** action, T' simulates the effect of executing an agent action selected according to π. That is, $T'((w, g), \textbf{noop}, (w', g))$ is equal to the probability that $T(w, \pi(w, g)) = w'$.

The **cost model** C' reflects the costs of agent and assistant actions in the MDP. For all actions a except for **noop** we have that $C'((w, g), a) = C(w, a)$. Otherwise we have that $C'((w, g), \textbf{noop}) = E[C(w, a)]$, where a is a random variable distributed according to $\pi(\cdot|w, g)$. That is, the cost of a **noop** action is the expected cost of the ensuing agent action.

The **observation distribution** μ' is deterministic and reflects the fact that the assistant can only directly observe the world state and actions of the agent. For the **noop** action in state (w, g) leading to state (w', g), the observation is (w', a) where a is the action executed by the agent immediately after the **noop**. For all other actions the observation is equal to the W component of the state, i.e. $\mu'(w', g) = (w', a)$. For simplicity of notation, it is assumed that the W component of the state encodes the preceding agent or assistant action a and thus the observations reflect both states and actions.

We assume an episodic objective where episodes begin by drawing initial POMDP states and end when arriving in a state (w, g) where w satisfies goal g. A policy π' for the assistant POMDP maps state-action sequences to assistant actions. The expected cost of a trajectory under π' is equal to our objective function $E[C(I, G_0, \pi, \pi')]$ from the previous section. Thus, solving for the optimal assistant POMDP policy yields an optimal assistant. However, in our problem setup the assistant POMDP is not directly at our disposal as we are not given π or G_0. Rather we are only given the environment MDP and the set of possible goals. As described in the next section our approach will approximate the assistant POMDP by estimating π and G_0 based on observations and select assistive actions based on this model.

4 Selecting Assistive Actions

Approximating the Assistant POMDP. One approach to approximating the assistant POMDP is to observe the agent acting in the environment, possibly while being assisted, and to learn the goal distribution G_0 and policy π. This can be done by storing the goal achieved at the end of each episode along with the set of world state-action pairs observed for the agent during the episode. The estimate of G_0 can then be based on observed frequency of each goal (perhaps with Laplace correction). Likewise, the estimate of $\pi(a|w,g)$ is simply the frequency for which action a was taken by the agent when in state w and having goal g. While in the limit these estimates will converge and yield the true assistant POMDP, in practice convergence can be slow. This slow convergence can lead to poor performance in the early stages of the assistant's lifetime. To alleviate this problem we propose an approach to bootstrap the learning of π.

In particular, we assume that the agent is reasonably close to being optimal. This is not unrealistic in many application domains that might benefit from intelligent assistants. There are many tasks, that are conceptually simple for humans, yet they require substantial effort to complete. Given this assumption, we initialize the estimate of the agent's policy to a prior that is biased toward more optimal agent actions. To do this we will consider the environment MDP with the assistant actions removed and solve for the Q-function $Q(a,w,g)$. The Q-function gives the expected cost of executing agent action a in world state w and then acting optimally to achieve goal g using only agent actions. We then define the prior over agent actions via the Boltzmann distribution. In our experiments, we found that this prior provides a good initial proxy for the actual agent policy, allowing for the assistant to be immediately useful. We update this prior based on observations to better reflect the peculiarities of a given agent. Computationally the main obstacle to this approach is computing the Q-function, which need only be done once for a given application domain. A number of algorithms exist to accomplish this including the use of factored MDP algorithms [Boutilier *et al.*, 1999], approximate solution methods [Boutilier *et al.*, 1999; Guestrin *et al.*, 2003], or developing specialized solutions.

Action Selection Overview. Let $O_t = o_1, ..., o_t$ be the observation sequence from the beginning of the current trajectory until time t. Each observation provides the world state and the previously selected action (by either the assistant or agent). Given O_t and an assistant POMDP the goal is to select the best assistive action according to the policy $\pi'(O_t)$.

Unfortunately, exactly solving the assistant POMDP will be intractable for all but the simplest of domains. This has led us to take a heuristic action selection approach. To motivate the approach, it is useful to consider some special characteristics of the assistant POMDP. Most importantly, the belief state corresponds to a distribution over the agent's goal. Since the agent is assumed to be goal directed, the observed agent actions provide substantial evidence about what the goal might and might not be. In fact, even if the assistant does nothing the agent's goals will often be rapidly revealed. This suggests that the state/goal estimation problem for the assistant POMDP may be solved quite effectively by just observing how the agent's actions relate to the various possible goals. This also suggests that in many domains there will be little value in selecting assistive actions for the purpose of gathering information about the agent's goal. This suggests the effectiveness of myopic action selection strategies that avoid explicit reasoning about information gathering, which is one of the key POMDP complexities compared to MDPs. We note that in some cases, the assistant will have pure information-gathering actions at its disposal, e.g. asking the agent a question. While we do not consider such actions in our experiments, as mentioned below, we believe that such actions can be handled via shallow search in belief space in conjunction with myopic heuristics. With the above motivation, our action selection approach alternates between two operations.

Goal Estimation. Given an assistant POMDP with agent policy π and initial goal distribution G_O, our objective is to maintain the posterior goal distribution $P(g|O_t)$, which gives the probability of the agent having goal g conditioned on observation sequence O_t. Note that since the assistant cannot affect the agent's goal, only observations related to the agent's actions are relevant to the posterior. Given the agent policy π, it is straightforward to incrementally update the posterior $P(g|O_t)$ upon each of the agent's actions. At the beginning of each episode we initialize the goal distribution $P(g|O_0)$ to G_0. On timestep t of the episode, if o_t does not involve an agent action, then we leave the distribution unchanged. Otherwise, if o_t indicates that the agent selected action a in state w, then we update the distribution according to $P(g|O_t) = (1/Z) \cdot P(g|O_{t-1}) \cdot \pi(a|w,g)$, where Z is a normalizing constant. That is, the distribution is adjusted to place more weight on goals that are more likely to cause the agent to execute action a in w.

The accuracy of goal estimation relies on how well the policy π learned by the assistant reflects the true agent. As described above, we use a model-based bootstrapping approach for estimating π and update this estimate at the end of each episode. Provided that the agent is close to optimal, as in our experimental domains, this approach can lead to rapid goal estimation, even early in the lifetime of the assistant.

We have assumed for simplicity that the actions of the agent are directly observable. In some domains, it is more natural to assume that only the state of the world is observable, rather than the actual action identities. In these cases, after observing the agent transitioning from w to w' we can use the MDP transition function T to marginalize over possible agent actions yielding the update,

$$P(g|O_t) = (1/Z) \cdot P(g|O_{t-1}) \cdot \sum_{a \in A} \pi(a|w,g) T(w,a,w').$$

Action Selection. Given the assistant POMDP M and a distribution over goals $P(g|O_t)$, we now address the problem of selecting an assistive action. For this we introduce the idea of an *assistant MDP* relative to a goal g and M, which we will denote by $M(g)$. Each episode in $M(g)$ evolves by drawing an initial world state and then selecting assistant actions until a **noop**, upon which the agent executes an action drawn from its policy for achieving goal g. An optimal policy for $M(g)$ gives the optimal assistive action assuming that the agent is acting to achieve goal g. We will denote the Q-function of

$M(g)$ by $Q_g(w,a)$, which is the expected cost of executing action a and then following the optimal policy.

Our first myopic heuristic is simply the expected Q-value of an action over assistant MDPs. The heuristic value for assistant action a in state w given observations O_t is,

$$H(w,a,O_t) = \sum_g Q_g(w,a) \cdot P(g|O_t)$$

and we select actions greedily according to H. Intuitively $H(w,a,O_t)$ measures the utility of taking an action under the assumption that all goal ambiguity is resolved in one step. Thus, this heuristic will not select actions for purposes of information gathering. This heuristic will lead the assistant to select actions that make progress toward goals with high probability, while avoiding moving away from goals with high probability. When the goal posterior is highly ambiguous this will often lead the assistant to select **noop**.

The primary computational complexity of computing H is to solve the assistant MDPs for each goal. Technically, since the transition functions of the assistant MDPs depend on the approximate agent policy π, we must re-solve each MDP after updating the π estimate at the end of each episode. However, using incremental dynamic programming methods such as prioritized sweeping [Moore and Atkeson, 1993] can alleviate much of the computational cost. In particular, before deploying the assistant we can solve each MDP offline based on the default agent policy given by the Boltzmann bootstrapping distribution described earlier. After deployment, prioritized sweeping can be used to incrementally update the Q-values based on the learned refinements we make to π.

When it is not practical to solve the assistant MDPs, we may resort to various approximations. We consider two approximations in our experiments. One is to replace the user's policy to be used in computing the assistant MDP with a fixed default user's policy, eliminating the need to compute the assistant MDP at every step. We denote this approximation by H_d. Another approximation uses the simulation technique of *policy rollout* [Bertsekas and Tsitsiklis, 1996] to approximate $Q_g(w,a)$ in the expression for H. This is done by first simulating the effect of taking action a in state w and then using π to estimate the expected cost for the agent to achieve g from the resulting state. That is, we approximate $Q_g(w,a)$ by assuming that the assistant will only select a single initial action followed by only agent actions. More formally, let $\bar{C}_n(\pi,w,g)$ be a function that simulates n trajectories of π achieving the goal from state w and then averaging the trajectory costs. The heuristic H_r is identical to $H(w,a,O_t)$ except that we replace $Q_g(w,a)$ with the expectation $\sum_{w' \in W} T(w,a,w') \cdot \bar{C}(\pi,w',g)$. We can also combine both of these heuristics, which we denote by $H_{d,r}$. Finally, in cases where it is beneficial to explicitly reason about information gathering actions, one can combine these myopic heuristics with shallow search in belief space of the assistant MDP. One approach along these lines is to use sparse sampling trees [Kearns *et al.*, 1999] where myopic heuristics are used to evaluate the leaf nodes.

5 Experimental Results

We conducted user studies and simulations in two domains and present the results in this section.

5.1 Doorman Domain

In the doorman domain, there is an agent and a set of possible goals such as collect *wood*, *food* and *gold*. Some of the grid cells are blocked. Each cell has four doors and the agent has to open the door to move to the next cell (see Figure 1). The door closes after one time-step so that at any time only one door is open. The goal of the assistant is to help the user reach his goal faster by opening the correct doors.

A state is a tuple $\langle s, d \rangle$, where *s* stands for the the agent's cell and *d* is the door that is open. The actions of the agent are to open door and to move in each of the 4 directions or to pickup whatever is in the cell, for a total of 9 actions. The assistant can open the doors or perform a **noop** (5 actions). Since the assistant is not allowed to push the agent through the door, the agent's and the assistant's actions strictly alternate in this domain. There is a cost of -1 if the user has to open the door and no cost to the assistant's action. The trial ends when the agent picks up the desired object.

In this experiment, we evaluated the heuristics H_d and H_r. In each trial, the system chooses a goal and one of the two heuristics at random. The user is shown the goal and he tries to achieve it, always starting from the center square. After every user's action, the assistant opens a door or does nothing. The user may pass through the door or open a different door. After the user achieves the goal, the trial ends, and a new one begins. The assistant then uses the user's trajectory to update the agent's policy.

The results of the user studies for the doorman domain are presented in Figure 2. The first two rows give cummulative results for the user study when actions are selected according to H_r and H_d respectively. The table presents the total optimal costs (number of actions) for all trials across all users without the assistant *N*, and the costs with the assistant *U*, and the average of percentage cost savings (*1-(U/N)*) over all trials and over all the users[1]. As can be seen, H_r appears to have a slight edge over H_d.

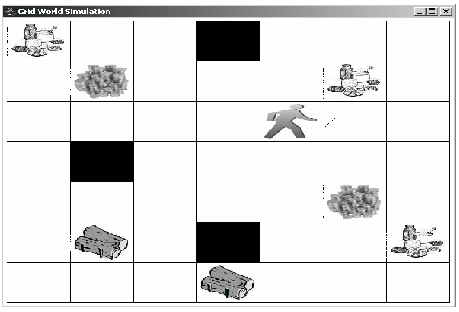

Figure 1: Doorman Domain.

To compare our results with a more sophisticated solution technique, we selected actions using sparse sampling [Kearns *et al.*, 1999] for depths d =2 and 3 while using b=1 or 2 samples at every step. The leaves of the sparse sampling tree are

[1]This gives a pessimistic estimate of the usefulness of the assistant assuming an optimal user and is a measure of utility normalized by the optimal utility without the aid of the assistant.

Heuristic	Total Actions N	User Actions U	Average $1-(U/N)$	Time per action
H_r	750	339	0.55± 0.055	0.0562
H_d	882	435	0.51 ± 0.05	0.0021
H_r	1550	751	0.543 ± 0.17	0.031
d = 2, b = 1	1337	570	0.588 ± 0.17	0.097
d = 2, b = 2	1304	521	0.597 ± 0.17	0.35
d = 3, b = 1	1167	467	0.6 ± 0.15	0.384
d = 3, b = 2	1113	422	0.623 ± 0.15	2.61

Figure 2: Results of experiments in the Doorman Domain. The first half of the table presents the results of the user studies while the lower half presents the results of the simulation.

evaluated using H_r. For these experiments, we did not conduct user studies, due to the high cost of such studies, but simulated the human users by choosing actions according to policies learned from their observed actions. The results are presented in the bottom half of Figure 2. We see that sparse sampling increased the average run time by an order of magnitude, but is able to produce a reduction in average cost for the user. This is not surprising, for in the simulated experiments, sparse sampling is able to sample from the exact user policy (i.e. it is sampling from the learned policy, which is also being used for simulations).It remains to be seen whether these benefits can be realized in real experiments with only approximate user policies.

5.2 Kitchen Domain

In the kitchen domain, the goals of the agent are to cook various dishes. There are 2 shelves with 3 ingredients each. Each dish has a recipe, represented as a partially ordered plan. The ingredients can be fetched in any order, but should be mixed before they are heated. The shelves have doors that must be opened before fetching ingredients and only one door can be open at a time.

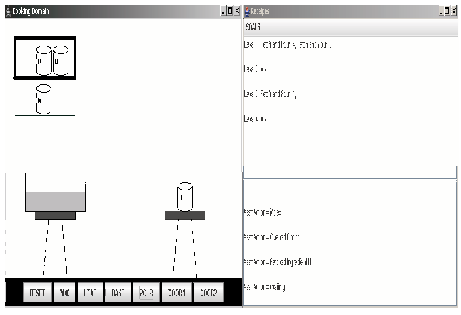

Figure 3: The kitchen domain. The user is to prepare the dishes described in the recipes on the right. The assistant's actions are shown in the bottom frame.

There are 8 different recipes. The state consists of the location of each of the ingredient (bowl/shelf/table), the mixing state and temperature state of the ingredient (if it is in the bowl) and the door that is open. The state also includes the action history to preserve the ordering of the plans for the recipes. The user's actions are: open the doors, fetch the ingredients, pour them into the bowl, mix, heat and bake the contents of the bowl, or replace an ingredient back to the shelf. The assistant can perform all user actions except for pouring the ingredients or replacing an ingredient back to the shelf. The cost of all non-pour actions is -1. Experiments were conducted on 12 human subjects. Unlike in the doorman domain, here it is not necessary for the assistant to wait at every alternative time step. The assistant continues to act until the **noop** becomes the best action according to the heuristic.

Since this domain has much bigger state-space than the first domain, both heuristics use the default user's policy. In other words, we compare H_d and $H_{d,r}$. The results of the user studies are shown in top half of the Figure 4. $H_{d,r}$ performs better than H_d. It was observed from the experiments that the $H_{d,r}$ technique was more aggressive in choosing non-noop actions than the H_d, which would wait until the goal distribution is highly skewed toward a particular goal. We are currently trying to understand the reason for this behavior.

Heuristic	Total Actions N	User Actions U	Average $1-(U/N)$	Time per action
$H_{d,r}$	3188	1175	0.6361 ±0.15	0.013
H_d	3175	1458	0.5371± 0.10	0.013
$H_{d,r}$	6498	2332	0.6379 ± 0.14	0.013
d = 2, b = 1	6532	2427	0.6277 ± 0.14	0.054
d = 2, b = 2	6477	2293	0.646 ± 0.14	0.190
d = 3, b = 1	6536	2458	0.6263 ±0.15	0.170
d = 3, b = 2	6585	1408	0.6367 ± 0.14	0.995

Figure 4: Results of experiments in the Kitchen Domain. The first half of the table presents the results of the user studies while the lower half presents the results of the simulation.

We compared the use of sparse sampling and our heuristic on simulated user trajectories for this domain as well (see bottom half of Figure 4). There is no significant difference between the solution quality of rollouts and sparse sampling on simulations, showing that our myopic heuristics are performing as well as sparse sampling with much less computation.

6 Related Work

Much of the prior work on intelligent assistants did not take a sequential decision making or decision-theoretic approach. For example, email filtering is typically posed as a supervised learning problem [Cohen *et al.*, 2004], while travel planning combines information gathering with search and constraint propagation [Ambite *et al.*, 2002].

There have been other personal assistant systems that are explicitly based on decision-theoretic principles. Most of these systems have been formulated as POMDPs that are approximately solved offline. For instance, the COACH system helped people suffering from Dementia by giving them appropriate prompts as needed in their daily activities [Boger *et al.*, 2005]. They use a plan graph to keep track of the user's progress and then estimate the user's responsiveness to determine the best prompting strategy. A distinct difference from our approach is that there is only a single fixed goal of washing hands, and the only hidden variable is the user responsiveness. Rather, in our formulation there are many possible goals and the current goal is hidden to the assistant. We note,

that a combination of these two frameworks would be useful, where the assistant infers both the agent goals and other relevant properties of the user, such as responsiveness.

In *Electric Elves*, the assistant is used to reschedule a meeting should it appear that the user is likely to miss it. Since the system monitors users in short regular intervals, radical changes in the belief states are usually not possible and are pruned from the search space[Varakantham *et al.*, 2005]. In a distinct but related work [Doshi, 2004], the authors introduce the setting of interactive POMDPs, where each agent models the other agent's beliefs. Our model is simpler and assumes that the agent is oblivious to the presence and beliefs of the assistant. Relaxing this assumption without sacrificing tractability would be interesting.

Our work is also related to on-line plan recognition and can be naturally extended to include hierarchies as in the hierarchical versions of HMMs [Bui *et al.*, 2002] and PCFGs [Pynadath and Wellman, 2000]. Blaylock and Allen describe a statistical approach to goal recognition that uses maximum likelihood estimates of goal schemas and parameters [Blaylock and Allen, 2004]. These approaches do not have the notion of cost or reward. By incorporating plan recognition in the decision-theoretic context, we obtain a natural notion of assistance, namely maximizing the expected utility.

There has been substantial research in the area of user modelling. Horvitz et.al took a Bayesian approach to model whether a user needs assistance based on user actions and attributes and used it to provide assistance to user in a spreadsheet application[Horvitz *et al.*, 1998]. Hui and Boutilier used a similar idea for assistance with text editing[Hui and Boutilier, 2006]. They use DBNs with handcoded parameters to infer the type of the user and computed the expected utility of assisting the user. It would be interesting to explore these kind of user models in our system to determine the user's intentions and then compute the optimal policy for the assistant.

7 Summary and Future Work

We described the assistant POMDP as a model for selecting assistive actions. We also described an approximate solution approach based on iteratively estimating the agent's goal and selecting actions using myopic heuristics. Our evaluation using human subjects in two game-like domains show that the approach can significantly help the user. One future direction is to consider more complex domains where the assistant is able to do a series of activities in parallel with the agent. Another possible direction is to assume hierarchical goal structure for the user and do goal estimation in that context. Our framework can be naturally extended to the case where the environment is partially observable to the agent and/or the assistant. This requires recognizing actions taken to gather information, e.g., opening the fridge to decide what to make based on what is available. Another important direction is to extend this work to domains where the agent MDP is hard to solve. Here we can leverage the earlier work on learning apprentice systems and learning by observation [Mitchell *et al.*, 1994]. The user's actions provide training examples to the system which can be used for learning.

Acknowledgements

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

[Ambite *et al.*, 2002] J. L. Ambite, G. Barish, C. A. Knoblock, M. Muslea, J. Oh, and S. Minton. Getting from here to there: Interactive planning and agent execution for optimizing travel. In *IAAI*, pages 862–869, 2002.

[Bertsekas and Tsitsiklis, 1996] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[Blaylock and Allen, 2004] N. Blaylock and J. F. Allen. Statistical goal parameter recognition. In *ICAPS*, 2004.

[Boger *et al.*, 2005] J. Boger, P. Poupart, J. Hoey, C. Boutilier, G. Fernie, and A. Mihailidis. A decision-theoretic approach to task assistance for persons with dementia. In *IJCAI*, 2005.

[Boutilier *et al.*, 1999] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *JAIR*, 11:1–94, 1999.

[Bui *et al.*, 2002] H. Bui, S. Venkatesh, and G. West. Policy recognition in the abstract hidden markov models. *JAIR*, 17, 2002.

[CALO, 2003] CALO. Cognitive agent that learns and organizes, http://calo.sri.com., 2003.

[Cohen *et al.*, 2004] W. W. Cohen, V. R. Carvalho, and T. M. Mitchell. Learning to classify email into speech acts. In *Proceedings of Empirical Methods in NLP*, 2004.

[Doshi, 2004] P. Doshi. A particle filtering algorithm for interactive pomdps, 2004.

[Guestrin *et al.*, 2003] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored MDPs. *JAIR*, pages 399–468, 2003.

[Horvitz *et al.*, 1998] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *In Proc UAI*, pages 256–265, Madison, WI, July 1998.

[Hui and Boutilier, 2006] B. Hui and C. Boutilier. Who's asking for help?: a bayesian approach to intelligent assistance. In *IUI*, pages 186–193, 2006.

[Kearns *et al.*, 1999] M. J. Kearns, Y. Mansour, and A. Y. Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *IJCAI*, 1999.

[Mitchell *et al.*, 1994] T. M. Mitchell, R. Caruana, D. Freitag, J.McDermott, and D. Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):80–91, 1994.

[Moore and Atkeson, 1993] A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103–130, 1993.

[Pynadath and Wellman, 2000] D. V. Pynadath and M. P. Wellman. Probabilistic state-dependent grammars for plan recognition. In *UAI*, pages 507–514, 2000.

[Varakantham *et al.*, 2005] P. Varakantham, R. T. Maheswaran, and M. Tambe. Exploiting belief bounds: practical pomdps for personal assistant agents. In *AAMAS*, 2005.