A Decision-Theoretic Model of Assistance

Alan Fern and **Sriraam Natarajan** and **Kshitij Judah** and **Prasad Tadepalli**
School of EECS, Oregon State University

Abstract

There has been 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 intelligent assistance. The objective is to observe a goal-directed agent and to select assistive actions in order to minimize the overall cost. We formulate the problem as an assistant POMDP where the hidden state corresponds to the agent's unobserved goals. This formulation allows us to exploit (partial) 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—one based on MDP planning, and another based on policy rollout. 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.

Introduction

The development of intelligent computer assistants has tremendous impact potential across many application 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 of interest. In this paper, we describe and evaluate a more comprehensive framework for intelligent assistants that captures and generalizes previous formulations.

We consider a model where the intelligent 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 robust 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 model-based approach is that it explicitly reasons about models of 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 approximate solutions.

A second contribution of this work 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.

Our third contribution is to evaluate our framework in two novel game-like computer environments, one a resource collection domain where the assistant acts as a "door man", and the second a kitchen domain where the assistant can assist in preparing meals. Our experiments on twelve human subjects indicate that in both environments the proposed assistant framework is able to significantly decrease user effort.

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.

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 directly 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 individual 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.

More 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, otherwise go back 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.

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 $I(w) \cdot G_0(g)$, 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 is only able to 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. $\mu'(w'|(w', g)) = 1$. For simplicity of notation, it is assumed that the W component of the state encodes the preceding agent or assistant action and thus the observations reflect both world states and actions.

Here we again assume an episodic objective where each episode begins by drawing an initial POMDP state and ends

when arriving in a state (w, g) such that w satisfies goal g. A policy π' for the assistant POMDP maps state-action sequences to assistant actions. For the assistant POMDP 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.

Selecting Assistive Actions

In this section, we first describe our approach to approximating the assistant POMDP. Next, we present our assistive action selection mechanisms.

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, particular for π. This slow convergence can lead to poor performance in the early stages of the assistant's lifetime. To alleviate this problem we propose a model-based approach to bootstrap the learning of π.

In particular, we assume that the agent is reasonably close to being optimal. This is not an unrealistic assumption in many application domains that might benefit from intelligent assistants. There are many tasks, for example, that are conceptually simple for humans, yet they require substantial effort to complete. Given this assumption, we will 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, Dean, & Hanks 1999), approximate solution methods (Boutilier, Dean, & Hanks 1999; Guestrin *et al.* 2003), or developing domain specific solutions.

Action Selection Overview. Let $O_t = o_1, ..., o_t$ be the sequence of observations gathered from the beginning of the current trajectory until timestep t. Each observation provides the current world state and the previously selected action (by either the assistant or agent). Given O_t and an (approximate) assistant POMDP our goal is to select the best assistive action according a 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 the policy π provided by the assistant POMDP reflects the true behavior of the agent. As described above, we use a model-based approach for bootstrapping our estimate of 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 the current distribution over goals $P(g|O_t)$, we now address the problem of selecting an assistive action. For this purpose, 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 $M(g)$ represents the best course of 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 a 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. In particular, the heuristic value for assistant action a in state w given previous observations O_t is given by,

$$H_1(w,a,O_t) = \sum_g Q_g(w,a) \cdot P(g|O_t)$$

and we select actions greedily according to H_1. Intuitively $H_1(w,a,O_t)$ 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 in computing H_1 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 & 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-value functions 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. One such approximation, which we will evaluate in our experiments, uses the simulation technique of *policy rollout* (Bertsekas & Tsitsiklis 1996) to approximate $Q_g(w,a)$ in the expression for H_1. 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. Our second heuristic $H_2(w,a,O_t)$ is identical to $H_1(w,a,O_t)$ except that we replace $Q_g(w,a)$ with the expectation $\sum_{w' \in W} T(w,a,w') \cdot \bar{C}(\pi,w',g)$.

Finally, we note that in cases where it is beneficial to explicitly reason about information gathering actions, it is straightforward to 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, Mansour, & Ng 1999) where myopic heuristics are used to evaluate the leaf nodes.

Experimental Results

In this section, we describe the results of user studies in two domains. Studies were conducted on 12 human subjects, where they were asked to achieve a given goal, which is hidden from the assistant. This particular protocol was chosen rather than letting the users choose their own goals to prevent them from changing their goals in mid-course, a behavior which was observed in earlier informal studies. The subject's and the assistant's actions were recorded. The ratio of the cost of achieving the goal with the assistant's help to the optimal cost without the assistant was calculated and averaged over the multiple trials for each user.

Grid World Domain

In the grid 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.

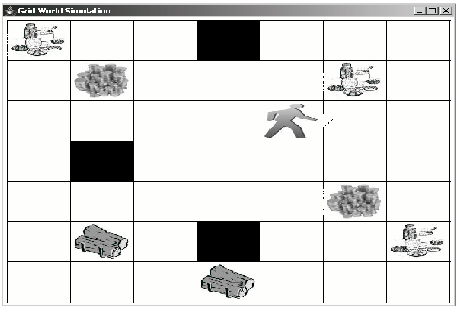

Figure 1: Grid Domain. The agent's goal is to fetch a resource. The grid cells are separated by doors that must be opened before passing through.

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. Experiments were conducted on 12 human subjects who are all graduate students in computer science. In each trial, the system chooses a goal and one of the two action-selection heuristics H_1 or H_2 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 agent 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.

User	Roll Out of User policies			Solving Assistant MDP		
	Total Actions	User Actions	Average (N_{UA} / N_{TA})	Total Actions	User Actions	Average (N_{UA} / N_{TA})
1	59	30	0.5142 ± 0.114	67	32	0.47 ± 0.17
2	79	34	0.433 ± 0.1211	34	16	0.45 ± 0.23
3	44	16	0.368 ± 0.077	79	40	0.512 ± 0.18
4	56	31	0.5435 ± 0.189	54	32	0.608 ± 0.13
5	60	30	0.5 ± 0.177	60	32	0.534 ± 0.14
6	69	28	0.398 ± 0.115	83	38	0.4525 ± 0.176
7	48	20	0.4219 ± 0.151	85	44	0.5197 ± 0.209
8	65	26	0.3912 ± 0.1156	111	50	0.447 ± 0.16
9	57	27	0.4944 ± 0.23	62	31	0.4916 ±0.14
10	71	32	0.46 ± 0.117	88	45	0.5083 ± 0.158
11	64	31	0.49 ± 0.19	78	36	0.46± 0.15
12	78	34	0.455 ± 0.168	81	39	0.484 ± 0.169
Total	750	339	0.452 ± 0.055	882	435	0.493 ± 0.046

Figure 2: Results of user studies in the Grid World Domain.

The results of the user studies for the grid world are presented in Figure 2. The table presents the total optimal costs (number of actions) for all trials without the assistant, and the costs with the assistant, and the average of percentage cost savings over all trials. As can be seen, both the methods reduce the cost to less than 50%. An omniscient assistant who knows the user's goal reduces the cost to 22%. This is not 0 because the first door is always opened by the user. In our experiments, if we do not count the user's first action, the cost reduces to 35%. This suggests that the assistant was very effective in reducing the cost for the user.[1].

Another interesting observation from the user study is that there are individual differences among the users. Some users always prefer a fixed path to the goal regardless of the assistant's actions. Some users are more flexible. This is the reason why the ratios in the table are not the same for all the users. From the survey we conducted at the end of the experiment, we learned that one of the features that the users liked was that the system was tolerant to their choice of suboptimal paths. Indeed, the data reveals that the system was able to reduce the costs by approximately 50% even when the users chose suboptimal trajectories.

[1]Human subject data is costly and we were unable to collect both "unassisted" and "assisted" data. We collected "assisted" human data and compared to the optimal "unassisted" policy, which gives a worst-case measure of assistant usefulness

Kitchen Domain

In the kitchen domain, the goal of the agent is to cook a dish. There are two shelves with 3 ingredients each. Each dish has a recipe, represented as a partially ordered plan. E.g., the ingredients can be fetched in any order, but should be mixed before they are heated. The applet on the left of Figure 3 shows the interface and the applet on the right gives the recipe. The bottom frame shows the latest set of actions that the assistant has taken. The shelves have doors that need to be open to fetch ingredients. In the figure, door 1 is open which is indicated by the darkened rectangle. At any instant, only one door can be open. Upon fetching an ingredient, the agent has to pour the ingredient into the bowl and then follow the recipe.

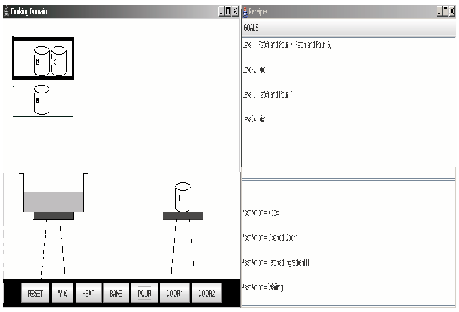

Figure 3: The kitchen domain. The user is to prepare the dishes described in the recipes on the right. The assistant's actions are shown in the bottom frame.

There are 7 different recipes to prepare. The state in this domain consists of the contents in the bowl, the door that is open, the mixing state and the temperature state of the bowl (heat/bake). 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. In addition, the user can replace an ingredient back to the shelf from which it was taken when the assistant fetches a wrong ingredient. The assistant can perform all the actions of the user except to pour the ingredients into the bowl or replace an ingredient back to the shelf. There is a cost of -1 for every non-pour user action.

Experiments were conducted on 12 human subjects. Unlike in the grid 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 action selection heuristic, and then the user takes an action. After the trial ends successfully, the assistant would use the trajectory to update the user's policy.

User	Roll Out of User policies			Approx Assistant MDP solver		
	Total Actions	User Actions	Average (N_{UA} / N_{TA})	Total Actions	User Actions	Average (N_{UA} / N_{TA})
1	109	59	0.543 ± 0.02	100	60	0.6 ± 0.056
2	90	54	0.6 ± 0.058	107	60	0.561 ± 0.05
3	111	60	0.54 ± 0.033	73	43	0.588 ± 0.058
4	99	58	0.588 ± 0.03	104	57	0.55 ± 0.04
5	74	41	0.55 ± 0.055	102	61	0.6 ± 0.064
6	125	71	0.576 ± 0.086	165	97	0.588 ± 0.06
7	99	62	0.633 ± 0.10	38	26	0.70 ± 0.09
8	66	43	0.65 ± 0.0525	102	61	0.597 ± 0.05
9	99	55	0.56 ± 0.06	103	61	0.59 ±0.063
10	93	59	0.635 ± 0.065	101	58	0.574 ± 0.062
11	114	64	0.564 ± 0.074	94	54	0.574 ± 0.0421
12	104	62	0.616 ± 0.116	85	53	0.62 ± 0.04
Total	1183	688	0.588 ± 0.038	1174	691	0.595 ± 0.038

Figure 4: Results of user studies in the Cooking Domain. The cost for the user is the number of the non-pour actions in the plan.

The results are shown in Figure 4. Similar to the other domain, the total cost of user's optimal policy, the total cost when the assistant is present, and the average ratio of the two are presented. There are no significant differences between the two methods of action selection.[2] The assistant reduces the number of user's actions to 60% in this domain. The performance was lower here than in the first domain because most of the recipes shared the same ingredients. Moreover, since there is a cost for fetching the wrong ingredient (since the human has to put it back) the assistant prefers to execute a **noop** rather than fetching a most likely ingredient.

Related Work

Our work is inspired by the growing interest in intelligent assistants. Some of this effort is focused on building desktop assistants that help with tasks such as email filtering and travel planning. Each of these tasks utilize different specialized technologies and lack an overarching framework. For example, email filtering is typically posed as a supervised learning problem (Cohen, Carvalho, & Mitchell 2004), while travel planning combines information gathering with search and constraint propagation (Ambite *et al.* 2002).

Our work is also related to on-line plan recognition. Currently popular approaches in this area are based on hierarchical versions of HMMs (Bui, Venkatesh, & West 2002) and PCFGs (Pynadath & Wellman 2000). Our framework can be naturally extended to include hierarchies. Blaylock and Allen describe a statistical approach to goal recognition that uses maximum likelihood estimates of goal schemas and parameters (Blaylock & Allen 2004). All these approaches, however, do not have the notion of cost or reward. By incorporating plan recognition in the decision-theoretic context, our formulation leads to a natural notion of assistance namely maximizing the expected utility.

There have also been personal assistant systems that are explicitly based on decision theoretic principles. For example, the COACH system is designed to help people suffering from Dementia by giving them appropriate prompts as needed in their daily activities (Boger *et al.* 2005). This problem is formulated as a POMDP and is solved offline. The system's actions are restricted to be communication actions. The goal of the human agent is fixed and the state is uncertain. We make the opposite assumptions that the goal of the agent is unknown, but the state is fully observed.

Researchers have leveraged the special properties of personal assistant systems to build efficient approximations for POMDPs that are solved offline. For example, since these systems monitor users in short regular intervals, radical changes in the belief states are usually not possible and are pruned from the search space (Varakantham, Maheswaran, & Tambe 2005). Neither exact nor approximate POMDP solvers are feasible in our online setting, where the POMDP is changing as we learn about the user, and must be repeatedly solved. They are either too costly to run (Boger *et al.* 2005), or too complex to implement as a baseline, e.g., Electric Elves (Varakantham, Maheswaran, & Tambe 2005). Our experiments demonstrate simple methods such as one-step look-ahead followed by roll-outs would work well in many domains where the POMDPs are solved online.

In (Doshi 2004), the authors introduce the setting of interactive POMDPs, where each agent models the other agent's belief state. Clearly, this is more general and more complex than ordinary POMDPs. Our model is simpler and assumes that the agent is oblivious to the presence and beliefs of the assistant, while the assistant models the agent's goal and the policy as explicit distributions. However since the agent's policy potentially changes in every step, the assistant POMDP should be solved in each step. This makes it difficult if not impossible to use off-the-shelf POMDP solvers because they take too long to compute the optimal policy. This is the reason we choose to focus on myopic solution techniques that nevertheless seem to make good if not optimal decisions most of the time.

Prior work on learning apprentice systems focused on learning from the users by observation (Mahadevan *et al.* 1993; Mitchell *et al.* 1994). This may be necessary, for example, when the the POMDP is not efficiently solvable even when the goal is fully known. Learning from observation can be easily incorporated into the our model by treating the user's actions as providing exploratory guidance to the system.

Summary and Future Work

In this work, we gave a general formulation of the problem of decision-theoretic assistance as a POMDP. Our model captures the uncertainty about the agent's goals and the policy, and the costs of the actions. Our algorithm consists of iteratively estimating the agent's goal and selecting appropriate assistive actions using myopic heuristics. We evaluated our framework on two domains using human subjects. The results demonstrate that the assistant was able to significantly reduce the agent's cost of problem solving.

[2] We did not solve the assistant MDP in this domain after every action. Instead, we approximated the assistant Q-values with the Q-values of the underlying MDP that ignored the assistant.

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. In these domains, the action selection might have to employ more sophisticated methods such as sparse sampling (Kearns, Mansour, & Ng 1999). 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 either the agent or the assistant or both. 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). Learning from observation can be easily incorporated into our model by treating the user's actions as providing exploratory guidance to the system. The system could update the values of these states using online reinforcement learning or incrementally learn a model and solve it offline.

References

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

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

Blaylock, N., and Allen, J. F. 2004. Statistical goal parameter recognition. In *ICAPS*, 297–305.

Boger, J.; Poupart, P.; Hoey, J.; Boutilier, C.; Fernie, G.; and Mihailidis, A. 2005. A decision-theoretic approach to task assistance for persons with dementia. In *IJCAI*, 1293–1299.

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

Bui, H.; Venkatesh, S.; and West, G. 2002. Policy recognition in the abstract hidden markov models. *JAIR* 17:451–499.

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

Cohen, W. W.; Carvalho, V. R.; and Mitchell, T. M. 2004. Learning to classify email into speech acts. In *Proceedings of Empirical Methods in Natural Language Processing*.

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

Guestrin, C.; Koller, D.; Parr, R.; and Venkataraman, S. 2003. Efficient solution algorithms for factored MDPs. *JAIR* 399–468.

Kearns, M. J.; Mansour, Y.; and Ng, A. Y. 1999. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *IJCAI*, 1324–1231.

Mahadevan, S.; Mitchell, T. M.; Mostow, J.; Steinberg, L. I.; and Tadepalli, P. 1993. An apprentice-based approach to knowledge acquisition. *Artif. Intell.* 64(1):1–52.

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

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

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

Varakantham, P.; Maheswaran, R. T.; and Tambe, M. 2005. Exploiting belief bounds: practical pomdps for personal assistant agents. In *AAMAS*, 978–985.