An agent architecture for intelligent information assistance

Jean Oh*, Felipe Meneguzzi*, Katia Sycara*, and Timothy J. Norman†
* Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
† Computing Science, University of Aberdeen, AB24 3UE, Scotland, UK

***Abstract*—Human users trying to plan and accomplish information-dependent goals in highly dynamic environments with prevalent uncertainty must consult various types of information sources in their decision-making processes while the information requirements change as they plan and re-plan. When the users must make time-critical decisions in information-intensive tasks they become cognitively overloaded not only by the planning activities but also by the information-gathering activities at various points in the planning process. We have developed the ANTicipatory Information and Planning Agent (ANTIPA) to manage information adaptively in order to mitigate user cognitive overload. To this end, the agent brings information to the user as a result of user requests but most crucially, it proactively predicts the user's prospective information needs by recognizing the user's plan; pre-fetches information that is likely to be used in the future; and offers the information when it is relevant to the current or future planning decisions. This paper describes a fully implemented agent of the *ANTIPA* architecture using a decision-theoretic user model, and reports preliminary user study results.**

I. Introduction

In a dynamic environment with prevalent uncertainty, users must consult various types of information in their decision-making processes while the information requirements change dynamically as users plan and re-plan. As a result, users who must make time-critical decisions in information intensive tasks are cognitively overloaded by the planning activities and the information requirements of the planning and re-planning.

For example, consider a military scenario where a commander must plan (or re-plan) a critical mission in a fast-changing real environment. Due to uncertainty and dynamics in the environment, the commander must constantly collect up to date information to ensure the success of the mission; reason about the feasibility of the current plan; and synchronize with other involved commanders (so that the overall plan is coherent). Here, information that the user must manage include intelligence reports, observations from the field, plan steps that must be executed, synchronization constraints, alerts, etc. In this context, we develop an information agent that can manage information adaptively so that the users can focus on planning activities without being overwhelmed by information-gathering activities.

In order to help the user to focus on planning tasks, the agent proactively *predicts* the user's prospective information needs by recognizing the user's plan; *optimizes* information gathering; and *presents* information in a way that alleviates the user's cognitive load. Our research hypothesis is that predictive information management based on a probabilistic model of user decision-making expedites the user's planning process and enables the user to explore a larger solution space; therefore, given a suitable user model, our agent assistant improves the quality of the plans generated by the user. To verify this hypothesis, we introduce a fully implemented information agent architecture known here as *ANTIPA*.

The agent's task is decomposed into four sub-problems: recognizing the user's current state; predicting the user's goals and plans; gathering information; and presenting information to the user. In this paper, we specifically focus on describing *ANTIPA*'s plan recognition algorithm (tackling the first two sub-problems), and omit the details of gathering and presenting information.

We take a decision-theoretic approach for our plan recognition algorithm. Other approaches use sequential decision-making models to design how an assistant agent should choose an optimal action based on its belief about the user's current state [1], [2]. In contrast, we use a decision-theoretic model only to represent how the user makes decisions, and use the model to predict how the user will behave in the future. The predicted user plan provides a set of goals for the *ANTIPA* agent, for which the agent plans and executes assistive actions asynchronously. This separation allows the *ANTIPA* agent to have a far richer planning capability when compared to other models where an assistant agent takes turns with the user in taking actions.

It is important to note that the goal of this research is not to guide the user in finding optimal planning solutions, but instead, the agent aims to optimize information management such that acquired information anticipates the user's information needs for planning decisions. As opposed to directing the user to make optimal decisions with respect to a certain objective (as in decision-support systems), we aim to design an agent that can maximize the support to help the user in making decisions based on her own criteria and judgement. From the user's perspective, independent decision making is crucial in many problem domains including military planning, educational support systems, and assistive living technologies for the disabled and the elderly.

This paper is organized as follows. First, we define the target problem in Section II, and introduce the *ANTIPA* agent architecture for intelligent information management in Section III. Next, we describe a decision-theoretic implementation of plan recognition within the architecture in detail in Section IV, followed by a discussion on related work in Section V.

Finally, we evaluate the approach through a game that is our abstraction of the information intensive environment in which our agent is designed to operate. We report promising initial results from user experiments in Section VI. We then conclude the paper with a discussion and future directions.

II. Problem Definition

We define information-dependent planning problems as a class of planning problems where a user (or a planner) must access various types of information sources to acquire current information that is required for executing certain actions. Here, in addition to domain-specific planning objectives, the user must also take the cost of getting information into consideration in selecting actions. Furthermore, the quality of information (that also depends on the source of information) affects the user's transition to another state after taking the action.

For instance, consider a student preparing for a final exam by reviewing selected topics covered in a semester. When a question is encountered, the student may search online for a quick answer by taking a risk that the answer found may be incorrect, or email the teacher and wait to get a generally more precise answer. The outcome of the student's *action* (to understand the concept) depends on the quality of information, which in turn depends on the source from which it has come. For example, by receiving high-quality information, the student's *state* regarding the understanding of a certain concept is more likely to *transition* from `not-learned` to `learned`, thus increasing the chance of getting a better grade (*reward*) in the final exam.

Given that the user is trying to solve an information-dependent planning problem, we design an agent that can adaptively identify and manage the user's information needs to facilitate the user's actions. The agent may not be able to directly observe the user's true states nor the actions that the user has taken; in this case, it must infer the user's state from a series of primitive sensory data known as *observations*. For instance, in the student example, possible observations include the keywords that the user types into search engines or a set of documents that the user opens.

III. The *ANTIPA* agent architecture

In order to address user cognitive overload in information-dependent planning problems described in Section II, we now introduce ANTicipatory Information and Planning Agent (*ANTIPA*), the intelligent information agent architecture for recognizing the user's plan simply by observing user behavior and managing relevant information requirements. Figure 1 depicts the high-level architecture of an *ANTIPA* agent where the agent processes are contained within the dashed box; the cloud represents the agent's observations; rounded boxes represent data structures; and rectangle stacks represent reasoning tasks. The basic interactions between the user and the agent are: the agent observes some of the user's planning activities; and the agent may present information to the user. At deployment time, the agent is supplied with two inputs: a *domain description* representing the user's planning problem

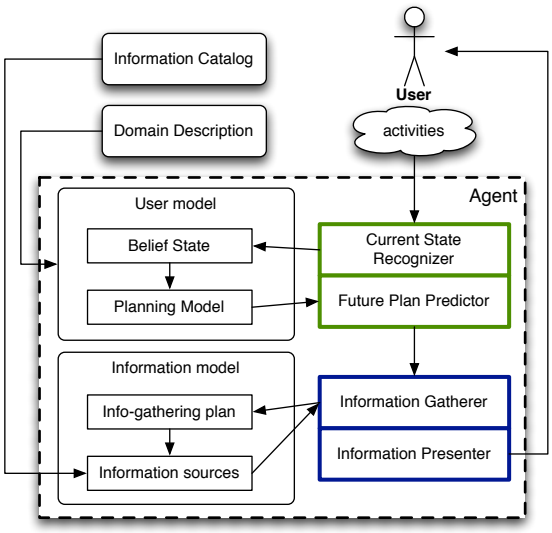

Fig. 1. The *ANTIPA* agent architecture.

(*e.g.*, state-based planning problems, plan libraries, workflow, or similar activity representations); and an *information catalog* that describes a set of properties of information sources from which the agent can retrieve information. These two inputs to the information agent are shown as the two rectangles at the top left of Figure 1.

As part of the process of deciding on collecting and presenting information, the agent's reasoning process tries to accomplish four main objectives. First, the agent must be able to identify the current state of the user from a sequence of observations on user activities. Second, the agent needs to predict the user's information needs that are changing dynamically through time. In order to accomplish this, the agent needs to identify the user's high-level goals and a set of planned actions to achieve these goals, in a process known as plan recognition [3]. If the agent can recognize the user's goals and plans, then the agent can infer the information needs associated with these planning activities. Third, the agent needs to construct a plan for collecting the information from various information sources. This plan must consider the tradeoff between obtaining high-priority information (of which the user is likely to make the most use) and satisfying temporal deadlines (i.e., certain information must be obtained before a specific time point to be useful). Finally, the agent must decide when to offer certain information to the user based on its belief about the user's current state. In order to accomplish these objectives, the *ANTIPA* architecture is composed of four main components as follows.

Current State Recognizer: The agent models the user's current state (which the agent cannot observe directly) as a probability distribution over a set of possible states, known as a *belief state*. When the agent perceives a new observation from the user interacting with an environment, Current State Recognizer updates its belief state such that the updated belief state best explains the observations. The updated belief state then triggers other components to adjust accordingly (*e.g.*, the agent can determine a set of information to present

immediately according to the current belief state).

Future Plan Predictor: Given a belief state, Future Plan Predictor identifies most likely plans from the current belief state and constructs a tree of action-nodes, known here as a *plan-tree*, representing a set of planning paths highly likely to be taken by the user. An action-node includes a *query* for the information that is required for the action (*e.g.*, a database query), the *priority*–the probability that the user will take the action, and a set of *constraints* (*e.g.*, a deadline constraint specifying the time by which the data must be retrieved). This plan-tree is then supplied to the Information Gatherer.

Information Gatherer: Given a plan-tree of predicted information-gathering tasks, Information Gatherer determines (or schedules) when and which information sources to use in order to satisfy the information needs of the user as well as coping with resource constraints (*e.g.*, network bandwidth) imposed by the problem domain; that is, the agent should not interfere with the user's planning activities by overconsuming computing resources. Initially, the information-gathering tasks are ordered by the priorities and the deadlines, ensuring not only the acquisition of the most useful information, but also a timely acquisition of data. In order to accommodate changing information requirements, Information Gatherer must optimize its current schedule incrementally to satisfy newer (thus more relevant) information-gathering constraints. The retrieved data is stored locally until used by Information Presenter.

Information Presenter: The agent directly interacts with the user through Information Presenter, which selects a subset of data from the locally cached data, and presents to the user at appropriate times. When to present which data is determined by the estimated user's future information needs. In order to avoid information overload, Information Presenter must only present data in temporal proximity to the actual need, with a sufficient time for the information to be useful for the action at hand. Additionally, Information Presenter must select an appropriate presentation format when offering information to the user. Finally, user feedback (*e.g.*, whether the presented information has been used) is collected and is provided for the agent as reinforcement in order to allow future improvements on the quality of supplied data.

IV. A Decision-Theoretic Approach

Towards the goal of recognizing user plan, we take a generative view where the user has a certain decision-making model; such that, if the agent manages to learn the model, then the agent can make the same decisions as the user in any given state. Our current approach is specifically focused on state-based planning problems such that a planning problem can be defined in terms of states, actions, reward, and a transition function. Here, we describe a formal representation of our decision-theoretic model, and the algorithm for predicting information needs using this model.

A. MDP-based User Model

We make two specific assumptions in modeling the user's decision-making process. First, we assume that the user's decision-making process respects the Markov property: the conditional probability of being in a certain state in the next time step depends only on the user's current state and not on any past states. Second, we assume that users will try to maximize the plan quality (while minimizing the action cost) by means of their local information and bounded reasoning capability.

Based on these assumptions, we use a *Markov Decision Process* (MDP) [4] to represent the user's planning process. An MDP is a specification of a sequential (discrete time) decision-making process for a fully observable environment with a stochastic transition model, *i.e.*, there is no uncertainty regarding the user's current state, but transitioning from one state to another is nondeterministic. The user planning objective modeled in an MDP is to create a plan that maximizes her long-term cumulative reward.

Formally, an MDP is represented as a tuple $\langle S, A, r, T, \gamma \rangle$, where S denotes a set of states; A, a set of actions; $r : S \times A \rightarrow \mathbb{R}$, a function specifying a reward (from an environment) of taking an action in a state; $T : S \times A \times S \rightarrow \mathbb{R}$, a state transition function; and γ, a discount factor indicating that a reward received in the future is less worth than an immediate reward. Solving an MDP generally refers to a search for a *policy* that maps each state to an optimal action with respect to a discounted long-term expected reward.

Because of our assumption that the agent can only partially observe the user's true states, Partially Observable MDP (POMDP) may appear more suitable to some readers such that the assistant agent takes the best assistive action according to the current belief state. However, such a POMDP model limits the agent to act in a sequential order by tightly coupling the agent's action selection with that of the user's. Here, we use an MDP to estimate how the user plans the future actions (when the user can fully observe her current state), so that the assistant agent can *plan* information-gathering actions for the predicted user plans (from the agent's current belief state) in order to satisfy the user's *future* information needs in a timely manner.

We now describe how to formulate an information-dependent planning problem as an MDP. Let I denote a set of information needs; $\Phi \rightarrow I$, a set of information sources that can satisfy information needs in I; S, a set of states describing the user's environment; A, a set of actions available from a state (that may be associated with certain information needs in I); $\tau : S \times A \times I \rightarrow \Re$, a domain-specific reward function; $c : I \times \Phi \rightarrow \Re$, an information-gathering cost function; and $T : S \times A \times I \times S \rightarrow \Re$, a function that defines how the user's state changes from the current state after taking an action that may require the gathering of certain information from the sources. Given information-dependent planning problem $g = \langle S, A, \tau, T, I, \Phi, c \rangle$, we construct a corresponding MDP $m = \langle S, A, \tau', T', \gamma \rangle$ as follows.

Since we assume that the user's planning objective is maximizing the reward while minimizing the information-gathering cost, we redefine reward function $\tau'(s, a)$ in state s after taking action a (which may require information i_a) as the net amount of reward by subtracting the expected cost $E[c(i_a)]$ of getting information i_a, such that: $\tau'(s, a) = \tau(s, a, i) - E[c(i_a)]$.

Since the cost of obtaining information is tied to the

choice of information source, the expected cost of information gathering depends on the user's selection strategy of information source. We estimate the selection strategy using a heuristic model where the user stochastically selects a source to maximize data accuracy as follows. Let $u_i(\varphi)$ denote the probability that the user selects source φ to get information i (such that $\sum_{\varphi\in\Phi} u_i(\varphi) = 1$); and let $\sigma(i,\varphi)$ denote a catalog lookup function that returns 1 if information i can be found in source φ, or 0 otherwise. Then, the probability $u_i(\varphi)$ is estimated proportionally to the data accuracy q_φ of source φ (which is specified in the information catalog), such that:

$$u_i(\varphi) = \frac{\sigma(i,\varphi)q_\varphi}{\sum_{\varphi'\in\Phi}\sigma(i,\varphi')q_{\varphi'}}.$$

Let $c_\varphi(i)$ denote the cost function of getting information i from source φ. Finally, the expected cost $E[c(i)]$ of retrieving information i can be expressed as follows:

$$E[c(i)] = \sum_{\varphi\in\Phi} u_i(\varphi)c_\varphi(i).$$

Lastly, the transition function of the original problem g is specified in terms of a conditional probability $T(s'|s,a,i)$ of the user transitioning from state s to another state s' after taking action a, given that information need i is satisfied. In order to express the transition probability in terms of state and action only, we apply Bayes rule as follows:

$$T'(s'|s,a) = \sum_{i\in I} p(i)T(s'|s,a,i),$$

where $p(i)$ denotes the probability of satisfying information need i. The probability $p(i)$ of satisfying information requirements can be obtained by summing up the probabilities of successful information acquisition from each source weighted by the probability $u_i(\varphi)$ of the user actually chooses the source:

$$p(i) = \sum_{\varphi\in\Phi} u_i(\varphi)v_\varphi q_\varphi,$$

where v_φ and q_φ denote the availability and the data accuracy of information source $\varphi \in \Phi$, respectively.

B. Current State Recognizer

To update a belief state, we use a variation of the *forward* algorithm [5], which we briefly sketch here. Let s_t denote the user's state at time t; $b = [b_1, ..., b_{|S|}]$, a belief state where $b(s) = p(s_t = s)$ is the belief probability of that the user is in state s at current time t; and $z_1, ..., z_t$, a series of observations from time step 1 through time step t. We assume that an initial probability $O(z|s)$ of the agent sensing observation z in state s is known (or it can be learned off-line). For each state $s \in S$, Current State Recognizer updates the probability of being in state s given a sequence of observations $z_1, ..., z_t$, denoted by $p(s_t = s|z_1, ..., z_t)$.

In order to compute this value efficiently, the algorithm instead utilizes the joint probability of that the user reaches state s at time t after collecting observations $z_1, ..., z_t$, denoted by $\alpha_s(t) = p(z_1, ..., z_t \wedge s_t = s)$ as follows. Given the first observation z_1 and initial belief state b, the initial α values at time step 1 can be computed for all states $s \in S$ as: $\alpha_s(1) = O(z_1|s_1 = s)b(s)$. As the agent receives new observation z, the α values are updated by recursively combining the previous alpha values of all incoming states with the probabilities of sensing the new observation in each state:

$$\alpha_s(t+1) = O(z|s)\sum_{s'\in S} T''(s',s)\alpha_{s'}(t),$$

where we estimate state transition probability $T''(s'|s)$ by combining state transition function $T'(s'|s,a)$ of the MDP user model and its optimal policy π (which can be computed using known algorithms such as *value iterations* [4]). By summing up transition probabilities $T'(s'|s,a)$ for all the actions dictated by policy π we get:

$$T''(s'|s) = \sum_{a\in A} \pi_s(a)T'(s'|s,a).$$

Finally, the belief state can be updated by normalizing the current alpha values using the following equation:

$$b(s) = \frac{\alpha_s(t)}{\sum_{s'\in S}\alpha_{s'}(t)}.$$

The belief state is updated whenever the agent receives a new observation, notifying Future Plan Predictor and Information Presenter accordingly.

C. Future Plan Predictor

We formulate the user's decision making as an MDP described in Section IV-A, so that the agent can predict the user's future plan by computing the optimal policy of the MDP. For solving an MDP, we use a well-known dynamic programming algorithm known as *value iteration* for simplicity, but more efficient algorithms can also be used instead. We construct a stochastic policy such that a policy specifies a probabilistic distribution over a set of available actions from each state (as opposed to a deterministic policy that maps a state to the best action). The intuition for using a stochastic policy is to allow the agent to explore multiple likely plan paths in parallel.

As an output, Future Plan Predictor constructs a tree-like plan segment in which a node contains a predicted user-action and a set of associated features. The features considered in our implementation consist of the action description (or an information query in case of information actions), the priority and the deadline of information need. The *priority* of the information need is specified using the probability representing the agent's belief that the user will select the action in the future. The *deadline* by which the data must be fetched is indicated by the depth of a node (in the tree structure) since the depth implies the number of time steps away from the current time to the future time step when the user will execute the action.

The algorithm builds a plan-tree by traversing the most likely actions (to be selected by the user) from the current belief state according the policy generated from the current MDP user model. First, the algorithm creates a root node

Algorithm 1 Future step prediction, where: node n; policy π; state s; normalization factor p

function PREDICT-FUTURE-STEPS(n, π, s, p)
  for all action $a \in A$ **do**
    $p' \leftarrow \pi(s, a)p$
    if $p' > \theta$ **then**
      $n' \leftarrow newTreeNode(a, p')$
      addChild(n, n')
      $s' \leftarrow$ sampleNextState(s, a)
      PREDICT-FUTURE-STEPS(n', π, s', p')
    end if
  end for
end function

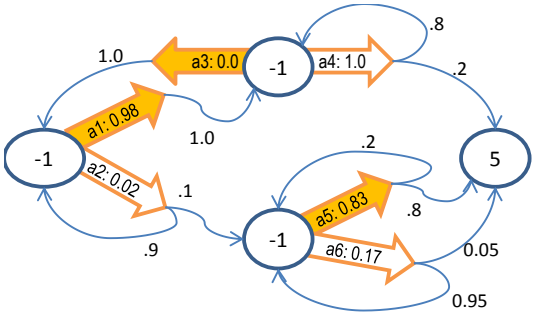

Fig. 2. An example of MDP user model and its optimal policy.

with probability 1 with no action attached. Then, according to the belief probability distribution over the set of states and to the MDP policy, likely actions are sampled such that the algorithm assigns higher priorities to those actions that are available from the states where the agent's belief is densely assigned and to those actions that lead to a better state with respect to the user's planning objective. The recursive process of predicting and constructing a plan tree from a state is described in Algorithm 1.

Note that the algorithm adds a new node for an action only if the agent's belief of the user selecting the action is higher than some threshold θ; actions are pruned otherwise. When this routine is called recursively, the probability of an action being selected is weighted so that the probabilities of child-nodes sum to the probability of their parent node.

Figure 2 shows an example of the MDP user model where a state is drawn as an oval with a reward written inside; an action as a boxed arrow (where information-dependent actions are shaded); and, state transition in directed edges annotated with transition probabilities. The optimal policy computed using discount factor 0.95 is written inside each action, *e.g.*, in the leftmost state, the policy prescribes to take action `a1` with probability 0.98 and action `a2` with probability 0.02. A plan tree constructed from this MDP is illustrated in Figure 3 where a tree edge is labeled with the probability of an action (on the right side of the edge) being selected according to the agent's belief. Note that actions `a3` and `a6` are pruned out (and thus drawn in dashed lines) when probability threshold is 0.01.

The resulting plan-tree represents a horizon of sampled actions for which the agent can prepare appropriate assistance.

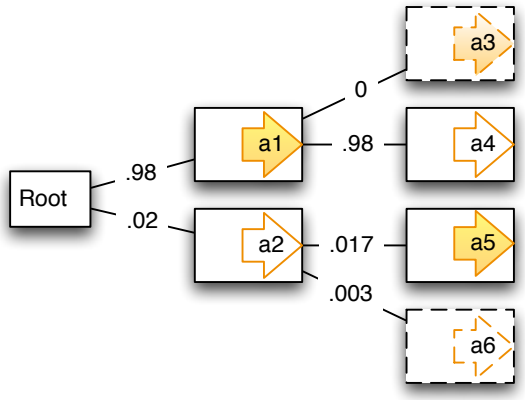

Fig. 3. A plan-tree predicted from MDP user model in Figure 2.

This predicted plan is provided to Information Gatherer to determine the information to acquire from various sources.

V. Related Work

Plan recognition has been studied in various fields: assistive technologies where assistant agents can guide a human user to execute a plan correctly [1]; cooperative multiagent problems where individual agents can infer the plans of other agents to synchronize their actions [6]; adversarial multiagent systems where an agent tries to figure out the intention of an adversary from observed actions [7]; intelligent user interface that can predict the next user action [8], and more can be found in a survey as in [3].

There has been a renewed interest in decision-theoretic approaches to plan recognition, shown in recent efforts in which planning algorithms are applied to recognize user plans without the need for constructing elaborate plan libraries of all possible plan alternatives [9]. Notably, researchers in cognitive science used an MDP model similar to ours to represent how a human predictor recognizes the plan of another actor by observing a sequence of the actor's current activities [10].

In an approach known as imitation learning (*a.k.a.* inverse optimal control), an expert demonstrates to an (apprentice) agent a set of (semi-) optimal decision-making examples during a training session, from which the agent tries to learn the (hidden) reward function that matches with the expert's decision making [11], [12], [13]. This approach is generally restricted to cases where a reward from a state linearly depends on a set of features of the state. Our current implementation does not have the learning capability for updating the MDP user model online; the learning capability will be sought in future work.

A POMDP-based approach was used in [1] to assist dementia patients, where the agent learns an optimal policy to take a single best assistive action in a belief state. In contrast, the *ANTIPA* architecture separates plan recognition from the agent's action selection (*e.g.*, gathering or presenting information), which allows the agent to asynchronously plan and execute multiple alternative information-gathering (or information-presenting) actions. Our work is thus focused

on how the agent can proactively assist the user after recognizing the user goals, *i.e.*, the agent can predict the user's most likely planning alternatives; subsequently, the agent can autonomously plan out assistive actions such as collecting relevant information.

Regarding information gathering, an approach exists for speculative plan execution in I/O-bounded planning problem domains [14]. In this approach, an agent uses a query classifier to speculate an outcome of a slow (computationally demanding) plan operator; continues to execute the next part of the plan using the guessed data; and verifies the speculated part of execution when the actual data from the source becomes available. In contrast, our information agent utilizes plan recognition techniques to predict the user's *future* plan and pre-fetches the information that the user is likely to need in the future.

In web browsing, the idea of prefetching is not new, but existing work focuses on prefetching based on the contents of web pages and the browsing patterns of common users as opposed to using the user's specific high-level plans, *e.g.*, Mozilla Link Prefetching or Google Web Accelerator. Many internet users, however, turn these options off because these approaches do not take user constraints into consideration such as network bandwidth constraints. We use a constraint-based planning technique for information gathering where user constraints and preferences can be seamlessly incorporated, which we will not discuss further since it is outside the scope of this paper.

VI. Experiments

We now describe the experimental setup, and discuss the user tests results.

A. Open-Sesame Game

For the purpose of our experiments, we designed a game, known here as Open-Sesame, that succinctly represents an information-dependent planning problem. The game consists of a grid-like maze where the four sides of a room in the grid can either be a wall or a door to an adjacent room; the user must enter a specific key code to open each door. The key codes are stored in a set of information sources; a catalog of information sources specifies which keys are stored in each source as well as the statistical properties of the source. Additionally, a browser-like interface is provided where the user can specify a door and an information source as inputs to retrieve the corresponding key code. The interface also allows the user to send multiple queries in parallel. Note that a door is closed when the user leaves the room, and thus the key code must be re-entered each time.

Here, depending on the user's planned path to the goal, the user needs a different set of key codes. Thus, the key codes to unlock the doors represent the user's information needs. When *ANTIPA* agent is enabled, the agent aims to predict the user's future path and prefetch the key codes that the user will need in the future.

We note that Open-Sesame is not meant to fully represent a real-world scenario, but rather to evaluate the ability of the architecture to predict information needs in a controlled environment and provide a preliminary validation of the architecture itself.

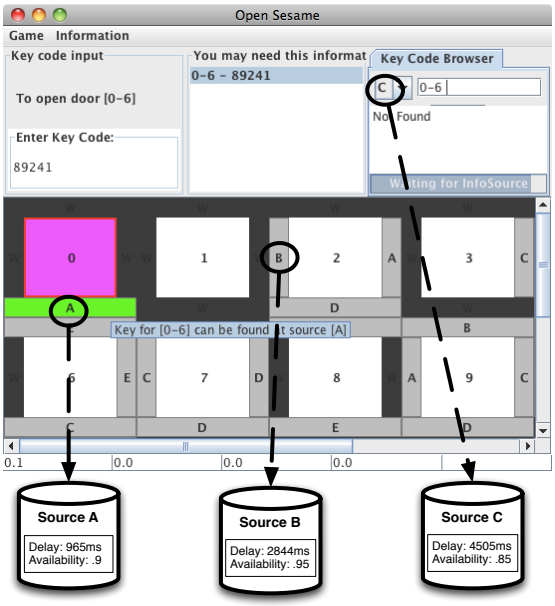

Fig. 4. Graphical user interface of Open-Sesame

The implementation of Open-Sesame is illustrated in Figure 4. In this example, the key codes are stored in three information sources A, B, and C. Information source A is available with .9 probability, and its average time delay is 965 ms. In the figure, room 0 is shaded, indicating the user is currently in the room. The three sides from room 0 facing north, east, and west are walls; and there exists a door that opens to room 6 to the south, the key code for which can be found in information source A.

B. Games and Settings

We created three Open-Sesame games: one 6×6 and two 7×7 mazes with varying degrees of difficulty. In each game, the key codes were distributed over 7 remote sources with varying source properties. Each room was painted with a randomly selected color from a set of 5 colors, which was the only type of observations for the agent (here, we purposely limited the agent's observation capability to simulate a partially observable setting). The starting position of the user was room 0 on the northwestern corner of the grid, and the objective of the game was to reach the room that has a star in it. The agent was given the map of a maze, the user's starting position, and the information catalog. During the experiments, each human subject was given 5 minutes of time to solve a game either *with* or *without* the agent assistance.

C. User Study Results

The results are summarized in Table I, which are based on the user tests on 7 human subjects who participated in 13 games. In the table, the total time measured the duration of a game; the game ended when the subject either has reached the

	Without agent	With agent
Total time (sec)	300	262.2
Total query time (sec)	48.1	10.7
Query time ratio	0.16	0.04
# of moves	13.2	14.6
# of steps away from goal	6.3	3

TABLE I
USER STUDY RESULTS

goal or has used up the given time. The results indicates that the subjects without agent assistance were not able to reach a goal within the given time, whereas the subjects with the agent assistance achieved a goal within the time limit in 6 out of 13 games.

The total query time refers to the time that a human subject has spent for information gathering, averaged over all the subjects under the same condition (*i.e.*, with or without agent assistance), and the query time ratio represents how much time a subject spent for information gathering relative to the total time. The agent assistance reduced the user's information-gathering time to less than $\frac{1}{4}$.

The number of moves that the user has made can be viewed as the user's search space in an effort to find a solution. That said, the agent assistant generally increased the search space. Although the sample size is too small to draw a statistically significant conclusion, these initial results are promising since they indicate that intelligent information management improves the user's performance with respect to the quality of solution (measured by the length of the shortest path to a goal from the subject's ending state).

VII. CONCLUSIONS

In this paper, we introduced an intelligent information agent architecture, *ANTIPA*, that proactively assists cognitively overloaded users through intelligent information management. The *ANTIPA* architecture presents a flexible framework that could possibly be customized for various types of intelligent assistant applications including adaptive learning support and assistive living technologies.

We described a decision-theoretic approach for plan recognition to identify the current planning progress conducted by the user, and to predict the user's future planning paths. Our current efforts aim at refining the information-managing components. Future work will include enhancing the information-gathering scheduler to take into consideration redundant information sources, as well as the design of a finer-grained process to reason about user cognitive overload. Moreover, we aim to develop online learning algorithms for estimating the user model such that the MDP user model is updated to best reflect the user's current behavior.

We empirically evaluated an implementation of the *ANTIPA* architecture through a user study. As part of evaluating our approach, we designed an information-gathering and navigation game, Open-Sesame. Despite its simplicity, Open-Sesame retains important characteristics of planning problem that requires information gathering in a concise form, which makes it suitable for evaluation purposes. Based on a set of preliminary user experiments, we conclude that the agent assistance significantly reduced the information-gathering time and enhanced the user performance during the games.

VIII. ACKNOWLEDGEMENT

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] 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 *Proc. IJCAI*, 2005, pp. 1293–1299.
[2] A. Fern, S. Natarajan, K. Judah, and P. Tadepalli, "A decision-theoretic model of assistance," in *Proc. of AAAI*, 2007.
[3] M. G. Armentano and A. Amandi, "Plan recognition for interface agents," *Artif. Intell. Rev.*, vol. 28, no. 2, pp. 131–162, 2007.
[4] R. Bellman, "A markov decision process," *Journal of Mathematical Mechanics*, vol. 6, pp. 679–684, 1957.
[5] L. Rabiner, "A tutorial on HMM and selected applications in speech recognition," *Proc. of IEEE*, vol. 77, no. 2, pp. 257–286, February 1989.
[6] G. Sukthankar and K. P. Sycara, "Robust recognition of physical team behaviors using spatio-temporal models," in *Proc. AAMAS*, 2006, pp. 638–645.
[7] D. Avrahami-Zilberbrand and G. A. Kaminka, "Incorporating observer biases in keyhole plan recognition (efficiently!)," in *Proc. AAAI*, 2007.
[8] P. Gorniak and D. Poole, "Predicting future user actions by observing unmodified applications," in *Proc. AAAI*, 2000, pp. 217–222.
[9] M. Ramírez and H. Geffner, "Plan recognition as planning," in *Proc. IJCAI*. Morgan Kaufmann Publishers Inc., 2009, pp. 1778–1783.
[10] C. Baker, R. Saxe, and J. Tenenbaum, "Action understanding as inverse planning," *Cognition*, vol. 31, pp. 329–349, 2009.
[11] A. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proc. ICML*, 2000, pp. 663–670.
[12] P. Abbeel and A. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. ICML*, 2004.
[13] B. Ziebart, A. Maas, J. Bagnell, and A. Dey, "Maximum entropy inverse reinforcement learning," in *Proc. AAAI*, 2008, pp. 1433–1438.
[14] G. Barish and C. A. Knoblock, "Speculative plan execution for information gathering," *Artif. Intell.*, vol. 172, no. 4-5, pp. 413–453, 2008.