A cognitive architecture for emergency response

Felipe Meneguzzi
Jean Oh,
Nilanjan Chakraborty,
Katia Sycara
Carnegie Mellon University
Pittsburgh, PA 15213 – USA
{meneguzz,jeanoh,nilanjan,katia}
@cs.cmu.edu

Siddharth Mehrotra
James Tittle
Agent Dynamics Inc.
Pittsburgh, PA 15213 – USA
siddharthmehrotra11@
gmail.com

Michael Lewis
University of Pittsburgh
Pittsburgh, PA 15260 – USA
ml@sis.pitt.edu

ABSTRACT

Plan recognition, cognitive workload estimation and human assistance have been extensively studied in the AI and human factors communities, resulting in many techniques being applied to domains of various levels of realism. These techniques have seldom been integrated and evaluated as complete systems. In this paper, we report on the development of an assistant agent architecture that integrates plan recognition, current and future user information needs, workload estimation and adaptive information presentation to aid an emergency response manager in making high quality decisions under time stress, while avoiding cognitive overload. We describe the main components of a full implementation of this architecture as well as a simulation developed to evaluate the system. Our evaluation consists of simulating various possible executions of the emergency response plans used in the real world and measuring the expected time taken by an unaided human user, as well as one that receives information assistance from our system. In the experimental condition of agent assistance, we also examine the effects of different error rates in the agent's estimation of user's stat or information needs.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Human Factors, Management

Keywords

Proactive Assistance, Emergency Management, Plan Recognition

1. INTRODUCTION

Planning for complex activities often involves consulting multiple information sources in order to reduce uncertainty associated with their decision making. As humans interleave planning, execution and re-planning, managing information to meet the changing requirements becomes a cognitively demanding task. Consequently, users who must make time-critical decisions are cognitively overloaded due not only to the planning activities but also to the information requirements of the planning and re-planning. In this context, we develop the *Anytime Cognition* (ANTICO) concept to assist cognitively overloaded users through an assistant agent.

Our approach to minimize user cognitive overload is to anticipate user's tasks before one actually needs to carry them out. By recognizing a user's plan for future activities, the assistant agent can act proactively to help the user balance her workload over time. ANTICO envisions a networked system of humans and software agents where the agents enhance human user performance by adaptively aiding in efficient and accurate decision-making. Assistance from ANTICO consists of proactive information gathering, and subsequent presentation of this information in a suitable form that takes into consideration a user's cognitive workload as well as the time available.

In order to transition the ANTICO concept to a real-world scenario, we develop a proof of concept application to assist a disaster response manager. This manager must deal with a chemical attack against a large civilian facility in a major US city, while dealing with uncertainty throughout the response. Uncertainty stems primarily in the diagnosis of the symptoms reported at the scene of the attack (and thus in the determination of the chemical used), and later from the various second-order effects. Here, the agent assists an emergency response manager who must make decisions under time-pressure, analyzing a stream of information arriving from various localized sources while keeping track of the big picture in order to effectively coordinate multiple agencies that must be directed to perform activities around the affected areas. Most crucially, since response managers must make decisions within tight deadlines, information needed for decision making must be presented in ways that facilitate quick action. For example, in the case of a chemical attack involving the sarin gas, the first responders should reach the site within $10 - 15$ minutes of attack (as per Department of Homeland Security (DHS) guidelines). Assuming some travel time required for the first responders to reach the event site, the operator has about 5 minutes to diagnose the attack and contact the relevant authorities, *e.g.*, Hazardous Materials (HazMat) unit and emergency medical services. Delays or misdiagnoses can be fatal not only for the population under attack but also for first responders.

In this paper, we introduce the Anytime Cognition (AN-

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

TICO) concept for a hybrid team of humans and software agents. ANTICO's contributions are threefold: first, we extend prior work on a proactive assistive agent architecture [5]; second, we deploy it in a concrete application domain; and third, we provide a simulation-based evaluation highlighting the circumstances in which gains could be obtained by our approach. We develop an emergency response scenario based on the standard disaster scenarios planning document [1], and present an application of ANTICO using this scenario, which has been fully implemented. Since potential gains from using ANTICO are closely associated to the accuracy of the agent in presenting relevant information, we evaluate the potential effectiveness of the approach through simulations of the assistance under various success rates for both intention prediction and information presentation.

We start the paper describing the emergency response scenario in Section 2, followed by a brief introduction of ADDL, an object-oriented domain description language in Section 3. We proceed to describe the agent architecture that supports the ANTICO concept in Section 4, and the software design and implementation using a step-through example in Section 5. This implementation is then evaluated in Section 6 by a simulation developed using the ADDL specification of the problem domain. We summarize related work in Section 7 and conclude the paper in Section 8.

2. SCENARIO DESCRIPTION

To demonstrate the applicability of the ANTICO approach to the real world, we develop a scenario based on the National Planning Scenarios created by the Department of Homeland Security (DHS)[1].[1] According to [1], planning the response for these scenarios encompasses 10 mission areas, which are more or less equivalent to phases in the response. ANTICO focuses on six of these areas, namely:

1. Emergency Assessment/Diagnosis;
2. Emergency Management/Response;
3. Incident/Hazard Mitigation;
4. Public Protection;
5. Evacuation/Shelter; and
6. Victim Care.

One major source of uncertainty in the early stages of many emergency response regards the identification of the nature of the disaster, i.e. diagnosing the emergency. This is particularly crucial when the disaster in question involves an invisible source of casualties, such as in the case of industrial accidents or attacks involving chemical, radiological and biological agents. In such cases, the wrong response might be more damaging than no response at all, since, for example, one may turn first responders into casualties if these are not prepared with appropriate protection. As a consequence, we have designed a response that considers the possibility of two different chemical agents from among the 15 possible disaster scenarios included in the DHS report, namely Sarin and the (fictitious) Yellow (Lewisite) chemicals. These two chemical agents were chosen because they have a small number of common symptoms and are colorless, creating a degree of initial uncertainty during the initial stages of the emergency response. Managing a response for a chemical attack involves complex workflow and information requirements for the operator. Although we have taken these two chemical agents as representative of the initial uncertainty, in a full deployment of ANTICO, dozens of similar chemical agents would need to be considered in a complete workflow.

The attack scenario utilized in the demo consists of the simultaneous release of multiple canisters of Sarin gas within a crowded public building in Washington DC, namely Union Station, by elements of a militant cell. Sarin is a chemical warfare agent classified as a nerve agent, and as such, is among the most toxic and rapidly acting chemical warfare agents currently known. These toxic agents act in a way similar to pesticides, but have a much stronger effect on human physiology. Sarin is expected to kill 95% of the people exposed to it in a confined environment, as well as affect a significant portion of the first responders, if not protected adequately. Since Sarin is a liquid that can be easily evaporated, once released, it will affect not only the people within the building, but it will also be released into the environment through the rooftop ventilation system within the building, creating a hazardous poison cloud that will affect people downwind from the original attack site.

In our simulation, the operator plays the role of an emergency management coordinator who receives messages from the field, tries to develop situation awareness for the developing incidents and dispatches/provides information to other human teams, such as police and medical services, to resolve the incidents. Notice that the operator has no prior knowledge of the actual chemical agent used in the attack. Since Sarin and Lewisite have somewhat similar symptoms, extra care must be taken that the correct chemical is identified and the appropriate responses are made, especially since the two chemicals have different time horizons and damaging effects. In order to provide assistance for operators in charge of coordinating response for these scenarios, we have developed detailed response workflows encoding the relevant actions, constraints and needed information for the various possibilities of these chemical attacks. To accomplish these tasks the operator must combine *pushed* information (*e.g.*, messages from units in the field or the ANTICO agent) with actively *pulled* information from various sources, such as contact telephone numbers of needed units (*e.g.*, HazMat) or reports of preparedness levels.

3. ANTICO DOMAIN DESCRIPTION LANGUAGE

ANTICO is designed as a generic agent architecture that can be applied to various problem domains. When developing a specific application, a software designer needs to engineer domain specific information such as user workflows and information sources. To facilitate this task, we designed the ANTICO Domain Description Language (ADDL) as an XML-based language for the problem domain. ADDL is not only a language to design a new application but also a well-structured medium used within the ANTICO components. An example fragment of ADDL is shown in Table 1.

An ADDL object consists of a set of user *activities*. An *activity* element specifies the significant factors for determining the user's current planning state, *i.e.*the activity currently being executed by the user. Each user activity has a unique

[1]`https://www.llis.dhs.gov/docdetails/details.do?contentID=13712`

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <anticoDomain>
3    <activity name="S0">
4      <observation name="START" prob="1"/>
5      <infoObject class="S0Plan/>
6      <transition>
7        <dest name="S1" prob=".25"/>
8            <dest name="S2" prob=".25"/>
9            <dest name="S3" prob=".25"/>
10           <dest name="S4" prob=".25"/>
11     </transition>
12   </activity>
13   <activity name="S1">
14     <observation name="yellow_symptom" prob=".7"/>
15     <observation name="yellow_symptom_loc" prob=".2"/>
16     <observation name="crowd_panic" prob=".1"/>
17     <infoObject class="S1Plan/>
18     <transition>
19       <dest name="S2" prob=".6"/>
20       <dest name="S15" prob=".4"/>
21     </transition>
22   </activity>
23     ...
24   <activity name="S48">
25     <observation name="traffic_info" prob=".2"/>
26     <observation name="click_police_info" prob=".56"/>
27     <observation name="click_dispatch_police" prob=".24"
           />
28     <infoObject class="S48Plan/>
29     <transition>
30       <dest name="S24" prob=".3"/>
31       <dest name="S26" prob=".7"/>
32     </transition>
33   </activity>
34     ...
35 </anticoDomain>
36 </xml>
```

Table 1: Fragment of an ADDL specification.

name and three main elements associated with it, *observations*, *infoObject* and *transitions*. In order for the agent to properly infer the planning state of the user, it must have a model of the observable features for each activity. An *observation* element represents an event observed at the specified user activity, associated with a *prob* attribute – the probability of the named observation occurring during the execution of a user activity. Although domain engineers might provide initial estimates of the probabilities of these observations, in realistic deployments, these values must be learned using data collected from previous episodes. Techniques for learning these probabilities include various Expectation-Maximization (EM) methods, such as the Baum-Welch algorithm [9]. The *infoObject* element points to the class that contains the information plan for a given activity. This class encodes the logic for extracting relevant information and basic instructions on its presentation to the user. The set of possible activity transitions are represented by the *transition* XML element. For each activity the transition element contains a list of possible destination activities represented by *dest* element with a unique name and probability of transition from the parent state. For example, Line 20 in Table 1 informs that the transition probability from activity $S1$ to activity $S15$ is 0.4.

In our deployment, we derive the parameters of the ADDL specification using a *Hidden Markov Model* (HMM) [9] with states representing the user activities. Here, an advantage of using a well structured language is that it could be extracted from any HMM description, and conversely, we can generate an HMM from the ADDL description, in order to apply learning algorithms over gathered data. When a new type of instruction is needed, ADDL can be extended to support the new instruction. Subsequently, appropriate handling methods must also be implemented to support the new instruction in the Information Presenter module.

4. ANYTIME COGNITION ARCHITECTURE

Concrete implementations of the ANTICO concept for proactive assistance are created following the generic ANTICO architecture. This architecture comprises multiple AI components including probabilistic plan recognition and intelligent information management. Figure 1 shows a modularized view of the ANTICO components and how those components are interconnected. The rectangles represent the main components; the third-party components are drawn in dotted lines; an ADDL document specifying a problem domain is provided as an input to the system; and the information object is the communication medium representing a user's information needs.

Here, we specifically focus on the following two desiderata for the assistant agent. First, the agent must be able to recognize a user's current and future activities. Second, the agent's interaction with the user must be unobtrusive and adaptive to user cognitive workload. The User Observer module is responsible for monitoring various parameters indicating a user's current activities and her environment. When a change is observed, the Intent Predictor module analyzes the new observation to identify the user's intention and makes predictions for the user's future activities according to a workflow model specified in ADDL. Subsequently, the Information Gatherer communicates with a set of information sources to meet the information requirements relevant to the predicted future user activities. Concurrently, the agent maintains an estimated user cognitive workload based on a set of observed temporal parameters in order to determine the appropriate level of detail in presenting information to the user.

The functions of each main component are described in the following subsections. We note that a more detailed description of the AI techniques underlying each of these components is provided in our previous work [5]. In this paper, we focus on the design, deployment and evaluation of the architecture.

4.1 User Observer

The User Observer module obtains and interprets user activities and messages that arrive from team members in the field. This module includes multiple *observer objects*, responsible for collecting and interpreting observations from different sources, *e.g.* user interface, input devices, external communication. Each observer object must implement the following three main methods: *monitor*, *interpret*, and *notify*. For instance, the mouse observer *monitors* a user's mouse clicks on a map, or scrolling up and down actions and *interprets* them into a set of predefined observation terms. The User Observer *notifies* the intent recognition module of new observations so that the agent's belief about the user's current and future intent can be updated accordingly.

4.2 Intent Predictor

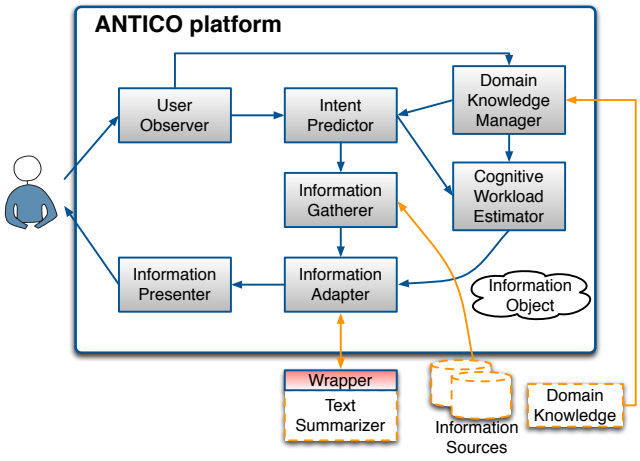

Figure 1: Architecture Overview

According to the workflow model (specified in ADDL), the Intent Predictor analyzes a series of observations collected from the User Observer to recognize the user's current state and the most likely sequences of future user activities. A user's current state (which the agent cannot observe directly) is represented as a probability distribution over a set of possible states, known as a *belief state vector*; and the most likely sequences of user actions are represented in a tree structure, a *plan-tree*, where each node of the tree specifies a user activity.

We update the belief state vector as follows. For each state $s \in S$, element $b(s)$ of belief state vector represents the probability of a user being in state s given a sequence of observations $z_1, ..., z_t$, denoted by $b(s) = p(s_t = s|z_1, ..., z_t)$. We can rewrite this probability as a fraction of seeing the particular state s after the series of observations over all possible cases as follows:

$$b(s) = \frac{p(z_1, ..., z_t \wedge s_t = s)}{\sum_{s' \in S} p(z_1, ..., z_t \wedge s_t = s')},$$

where $p(z_1, ..., z_t \wedge s_t = s)$ can be efficiently computed using a dynamic programming technique as shown in [9]. For instance, after observing the user opening an event message and clicking on a certain area of map, the agent updates its belief state to reflect that the user is assessing the event.

The updated belief state triggers two agent functions. First, the belief state is passed to the Information Presenter to determine if the agent has relevant information to present to the user. Second, the agent recomputes the plan-tree of predicted user activities likely to follow from the updated current user state, using a sampling method. Based on the assumption that the user will act according to the workflow model, a node is added for each action such that the information requirements associated with actions with higher expected utilities is assigned higher *priorities*. We note that the user's choice of action is not assumed to be necessarily optimal, but rather close to optimal, as has been shown by our previous work [5]. In addition, an activity is associated with the task *deadline* constraint specified in the workflow model, by which the data must be fetched since the user will need the information before executing the action.

As a result, a plan-tree node contains a predicted user-action (*e.g.*, Dispatch ambulance), *queries* specifying associated information requirements (which is annotated in the workflow model), *priority*, and *deadline*. For instance, based on the belief state that there is a toxic gas attack in a populous region, the agent recognizes that the user will need contact information of HazMat group as well as police for crowd control and orderly evacuation. The updated plan-tree is then supplied to the Information Gatherer to revise information gathering plan.

4.3 Cognitive Workload Estimator

The Cognitive Workload Estimator (CWE) uses inputs from the cognitive workload model[2] and the *deadline* specified in the workflow model. We use a queuing network based model for computing the workload of the user [3, 6, 7]. A standard queue-based model of mental workload treats the user's attention as the server and tasks as jobs (see [10] for details). As jobs arrive they are placed on the queue. Typically some simple discipline such as first in first out (FIFO) or priority-FIFO is used to manage the queue. The user's capacity is considered to have been exceeded when jobs on the queue exceed the user's ability to process them while avoiding tardiness. Based on the predicted plan-tree, the arriving tasks are determined, and the CWE maintains a running estimate of the probability that an operator will fail to complete a task by deadline. In the prototype implementation, we use a binary indicator such that the user cognitive workload is `overloaded` if the estimated probability of activity failure exceeds a certain threshold. Otherwise, the workload estimator returns `normal`.

4.4 Information Gatherer

Given a plan-tree of predicted information-gathering tasks, the 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; specifically, the agent should not interfere with the user's planning activities by over-consuming 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 to meet the deadline constraints. To accommodate changing information requirements, the 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 the Information Presenter.

4.5 Information Adapter

The Information Adapter determines the level of detail when presenting the data received from the Information Gatherer to the user, considering both the user cognitive workload and the estimated time available to the user for the activity. Obtaining precise quantitative relationships between cognitive workload of a user and the information content in a document that can be processed by her is an open problem. Thus we design the Information Adapter to present the information at various levels of granularity so that the user has a choice of the kind of content she wants

[2]The model development is outside the scope of this paper.

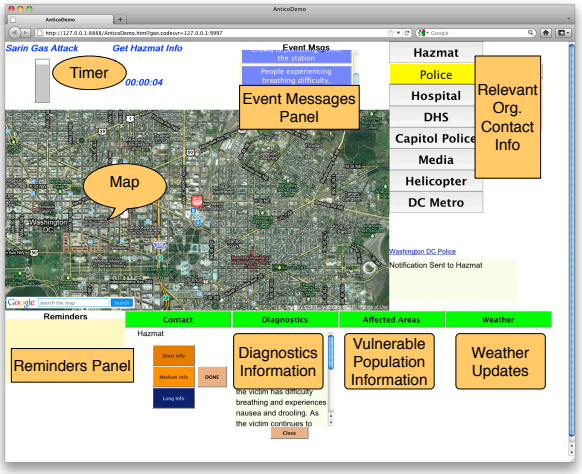

Figure 2: User interface for emergency response managers

to read. In the current implementation, we classify information sources according to the granularity of data stored in the sources.

4.6 Information Presenter

Given a belief state representing the user's current state, the Information Presenter selects relevant information in the local storage and presents it to the user. In order to avoid information overload, the Information Presenter must only present data in temporal proximity to the actual need, with a sufficient time for the information to be intelligible to the user to meet her decision deadlines for the action at hand. 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.

5. APPLICATION DESCRIPTION

This section illustrates the prototype application of ANTICO in the emergency response scenario described in Section 2 using a working example.

5.1 Graphical User Interface (GUI)

Figure 2 shows a snapshot of the operator interface annotated to explain the purpose of each panel. To display an interactive map, we integrate Google Maps[3] in our GUI.

The messages arriving from the field or other groups are shown in the *Event Messages* panel. In a real situation, these messages will arrive to the agent from actual sources (*e.g.* electronic messages from 911 operators). Here, the messages are introduced into the environment in order to simulate the real-time scenario. The time left to deal with the overall emergency situation is displayed in the top left part of the interface. The panels in the bottom of the interface display the following types of information:

- *Contact* information for organizations employed in emergency situations, *e.g.*, fire stations, local police, and facility security forces;
- *Diagnostics* for particular events;
- *Affected areas* and *vulnerable populations* such as schools and nurseries; and
- *Weather*: Weather information is useful since (a) it may affect the ability of response units to get to the emergency site, and (b) for certain types of events, for example during chemical attacks, winds can distribute the toxic chemical to large downstream areas, affected areas and vulnerable populations.

In addition, the *Reminder* panel (the leftmost in the bottom) provides reminders to the operator regarding special equipment or required capabilities for operation units. For example, the unit that evacuates vulnerable populations requires special training for evacuating handicapped people. Note that the user may access information sources directly at any time using the organization panel on the right.

Finally, although not necessarily part of the interface seen by a human operator, our implementation also includes a "debugging" interface (shown in Figure 3) that allows one to visualize the internal belief state of the information agent. This visualization shows the workflow currently being used by the agent, including all activities and transitions between activities, as well as the agent's current belief state expressed as probabilities associated with each user activity. These probabilities represent the degree of belief possessed by the agent that a user is currently carrying out the activity in the workflow. Thus, the excerpt of a workflow shown in Figure 3 shows that the agent believes with .88 probability that the user is currently carrying out the *Dispatch Ambulance* activity in the Sarin partition of the workflow. At the same time the agent believes with .12 probability that the user is carrying out the same activity in the Yellow partition of the workflow (reflecting a small degree of uncertainty over the diagnostics performed earlier).

5.2 Working example

To illustrate the working of ANTICO agent, we use a simple example to describe in detail what the user sees in the GUI, what the agent performs in the background, and how the user makes decisions.

During peak hour at Union Station in Washington DC, a chemical attack takes place. The expected size of crowd in the proximity of attack is approximately 20,000 with heavy traffic in the roads surrounding the station. For this particular scenario, we have created a workflow containing two similar partitions: one for responding to a Sarin Gas Attack; and one for responding to a Yellow Gas Attack. Although at a higher level these workflows are quite similar, the specific information and reminders that will be provided for the user differ, since the symptoms and treatment for these two chemical agents are different. After reading the event messages regarding toxic gas, the user clicks on an area in the map. This observation triggers the Intent Recognizer module and the agent's belief state indicates that the user is determining the type of gas whether it is Sarin or Yellow. At this point, with limited information, the intent recognizer will have probabilities distributed more or less evenly among the states in the various partitions of the workflow. This reflects the uncertainty at the diagnostics phase of the response, since Sarin and Yellow have a certain amount of similar symptoms. Thus, as represented in the workflow of

[3] `http://code.google.com/apis/maps/`

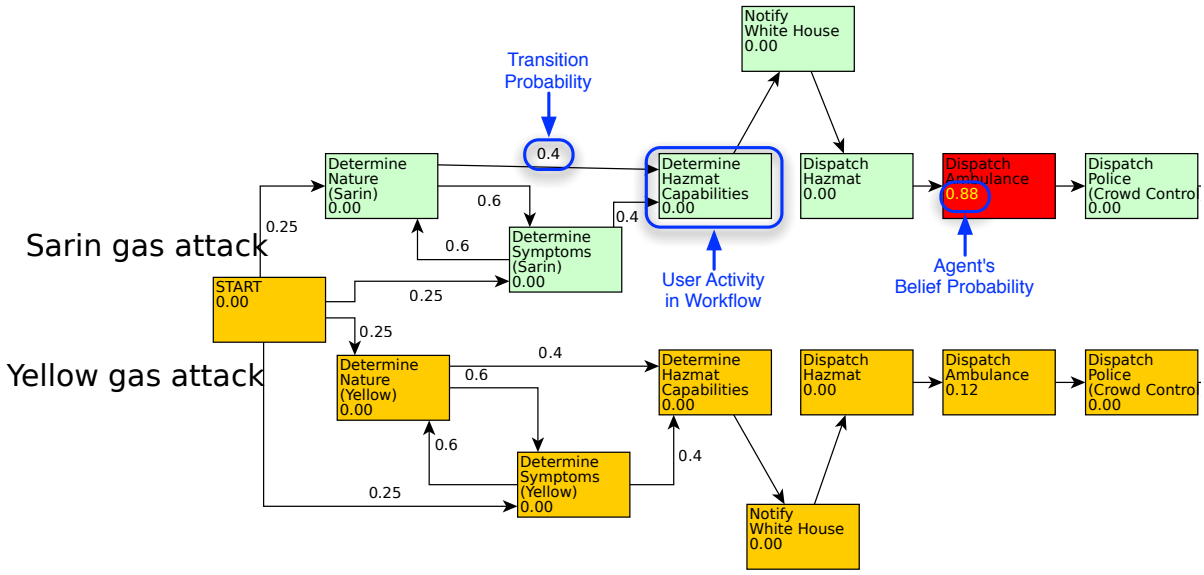

Figure 3: A fragment showing the user workflow model

Figure 3, the first activities the agent infers that a user will be carrying out relate to the determination of the nature of the attack and the symptoms reported from the scene. Accordingly, the type of information that the agent provides to the user at this point include a description of the symptoms of these chemical agents. Upon receiving a new set of event messages the agent's belief about Sarin gas is corroborated, and the agent immediately prefetches the diagnostics information of Sarin gas. Moreover, as the user peruses this information (which the agent detects as the user interacts with the information interface), the agent concludes that the user will soon want to dispatch a Hazardous Materials unit to the scene. Thus, the HazMat information nearby the affected area is retrieved.

At the same time, the agent presents HazMat information to the user as follows. Based on the workflow model, the optimal time for using HazMat information is calculated as two minutes. ANTICO presents the HazMat information in three levels of granularity (short, medium and long) classified by content size, and suggests the a level of granularity compatible with the predicted cognitive workload.

Figure 3 shows a visualization of the workflow available to the agent. Each activity is represented by a rectangle in the graph. Activities are connected by arrows in different directions; the numbers along the arrows represent the probabilities of the next activities. The number below each activity represents the estimated probability of the user performing the activity, *e.g.*, from the the agent's current belief state, the operator is mostly likely to be dealing with a Sarin gas attack. It also predicts that the operator's next action should be to "Dispatch Police" for crowd control, within the Sarin gas branch of the workflow.

Based on the actions taken by the user with the current information on the interface, the agent updates its belief state and keeps presenting the required information to the user in a new state within the time constraints imposed by the new events. For instance, as the emergency simulation progresses, ANTICO brings up information for live weather and traffic updates, vulnerable population and hospitals near the emergency site[4].

6. EVALUATION

In real disaster scenarios within the United States, emergency management is conducted by following an *Emergency Operations Plan* (EOP). Each major urban center in the US is expected to have EOPs available to key civilian personnel involved in managing disaster response. Consequently, when responding to a disaster by following the EOP, an unaided human user must look up the required information in the EOP and read a number of pages of text, for each step of the plan. In this setting, ANTICO proactively looks up the required information on behalf of the user, and summarizes this information to a degree appropriate with the time available and the level of cognitive workload experienced by the user. One of the key research questions regarding the effectiveness of ANTICO is the degree to which the agent can shorten *reaction time*. More importantly, we wanted to study how sensitive this time gain is affected by the accuracy with which the intent predictor infers the user current intention. Intuitively, if the agent always infers the user's current intention correctly and presents the correct information in summarized form, a human user should see large gains in terms of reaction time. Conversely, if the agent always misses the current intention and displays incorrect information, a user not only must refer to the EOP document, but also suffer the time penalty of reading the irrelevant information.

Given the difficulty in obtaining access to trained emergency management personnel, we have devised a simulation of a user managing an emergency scenario following the

[4]ANTICO demo video: `<omitted>`

workflow illustrated in Figure 3. This simulation allows us to evaluate the potential effectiveness of the ANTICO concept under various hypothetical error rates by the agent. The simulation consists of random walks through the workflow, following its transition probabilities, while accumulating the time taken by a human user to read the information needed to complete the task. Since we consider the amount of information actually read by the user during emergency management to be the main driver for the time spent carrying out a task, the main parameters of each step in the simulation are the best and worst-case scenario for the number of pages required to be read to complete an activity. Each activity in the workflow is associated with a particular section (or chapter) within the EOP document[5], and the amount of information needed at each task varied from none (for tasks where the emergency manager is expected to know the information) to six pages. In order to estimate the expected time spent by a human user reading this information, we took the standard average of 250 words per page, and used reading rates obtained from the human factors literature [4] that state that the mean number of *words per minute* (WPM) for human readers is 290 with a standard deviation of 49.3 WPM. As a result, in our simulation each task takes a number of seconds equal to $time(t,h) = \frac{P_t * 250}{W_h * 60}$ to complete, where: *i)* P_t is the number of pages *actually read* to complete task t, sampled from a uniformly random distribution between 0 and the size of the EOP section associated with task t; and *ii)* W_h is the reading rate of a human h, sampled from a gaussian random distribution. When the user is assisted by the agent, P_t varies depending on the accuracy of the agent. If the agent correctly predicts the information needs of the user, and the information presented is at the right level of granularity, P_t takes a random value between 0 and the values in the set $\{0.25, 0.5, 2\}$, representing the amount of information, in pages, summarized by the agent to *low*, *medium* and *high* levels of granularity. Conversely, if the agent has incorrectly predicted the user's information needs, not only must the user read the irrelevant information presented by the agent, it must also revert to consulting the EOP, and read the appropriate section, as if unassisted. Thus, when the agent fails in providing relevant assistance, the presented information becomes a penalty to the time taken by the user.

We generated 100000 samples with the simulator measuring the amount of time taken to execute the emergency management workflow for both the agent-assisted and unaided user. Within this set, subsets of 10000 samples were taken for assistance with an accuracy probability p varying from 0 to 1.0 in 0.1 increments. Using the resulting times of our simulations, we calculated the performance ratio between the agent-assisted and the unaided user. These results are illustrated in the graph of Figure 4.a, which shows the various accuracy values along the X axis, as well as the performance ratios (with standard deviation) along the Y axis. This performance ratio represents the fraction of time taken by the agent-assisted user to complete the workflow in relation to the unaided user, consequently, if the performance ratio is .5, the assisted user took half of the time taken by the unassisted user to complete the exact same sequence of activities. As expected, the time taken by a user to complete

[5]We had access to the table of contents of the EOP for a major US city, from which data size estimates were derived.

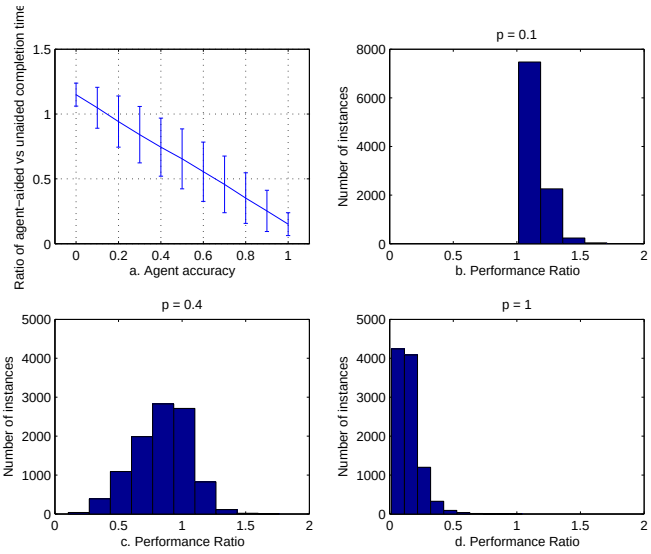

Figure 4: Simulation analysis

the tasks diminishes monotonically as the agent's intention recognition improves, at around 0.2 accuracy, the aided user starts performing similarly to the unaided user. Furthermore, we illustrate in more detail the specific number of samples associated with each performance ratio in the histograms of Figure 4.b-4.d for p equal to 0, 0.4 and 1. Bars to the left side of each histogram show samples in which the agent led to improved performance. Notice that at $p = 0.1$ the user's performance tends to be worse than the unaided user (since most samples have a ratio greater than 1). However, already at $p = 0.4$, most of the samples indicate an improvement in user performance.

7. RELATED WORK

In this section, we review three state of the art emergency response systems used today and point out the differences.

7.1 Google Crisis Response

Google *crisis response manager*[6] is an online tool that collects fresh high-resolution imagery plus other event-specific data, and then publishes this information on a dedicated landing page. Different users are also allowed to add information related to the disaster at a given geographic location, making this tool a wiki-like disaster response tool. Although having updated information is an important element of effective disaster response, this kind of application can easily overwhelm a responder by a deluge of information coming from user updates. On the one hand, the plethora of information can result in the operator spending too much time in obtaining and absorbing the information; thus leaving less time to accomplish coherent response tasks. On the other hand, spending too little time in acquiring the information may lead to inferior task performance. Whereas Google crisis response aggregates large amounts of data, possibly jeopardizing user's responsiveness, ANTICO avoids this problem by monitoring the user's cognitive workload and providing

[6] `http://www.google.com/crisisresponse/`

an amount of information proportional to the user's ability to deal with it.

7.2 WIPER

WIPER provides emergency response managers with an integrated system that detects possible emergencies from cellular communication data, attempts to predict the development of emergency situations, and provides tools for evaluating possible courses of action in dealing with emergency situations [8]. WIPER conveys three distinct pieces of information to responders via a web-based console:

- near-real time information on the location of cell phone users in an area, plotted on a GIS-based map of the area;
- potential anomalies, such as traffic jams, roving crowds and call patterns indicative of a crisis; and
- customized mitigation strategies, such as potential evacuation routes or barricade placement, suggested by computer simulations.

In order to provide such unobtrusive aid in context and in a timely manner, ANTICO signals to the user choices of information granularity that are likely to satisfy the effectiveness/timeliness tradeoff. WIPER, on the other hand, is an assistance system that helps a user plan to deal with a disaster. However, similarly to Google crisis response, WIPER does not deal with user information overload, and can provide too much information to the user. Thus, WIPER's capabilities for information gathering could be integrated in ANTICO's information gatherer module.

7.3 SOFER

Instead of providing proactive assistance, the Simulation-based Optimization of Fire and Emergency Response (SOFER) system [2] provides guidance on optimal allocation and placement of disaster response units to satisfy a set of service quality and availability criteria. SOFER uses a history of 911 requests to simulate future requests and determine how to distribute existing resources and the need to deploy additional resources in order to meet specific optimization criteria. It does not, however, provide on-demand assistance as disasters are occurring, but could be used in the context of an assistant to provide deployment suggestions for a user having to deal with multiple simultaneous disaster situations. Although SOFER is not an assistant *per se*, it provides optimal resource placement strategies that could be used to help mitigate a user's cognitive workload by suggesting deployment configurations as a disaster evolves.

8. CONCLUSIONS AND DISCUSSION

In this paper, we introduced the Anytime Cognition (ANTICO) concept, a generic information assistance architecture that employs probabilistic plan recognition techniques and a model of cognitive workload to proactively gather and present information to a human user such that the user is able to absorb the given information in order to perform the activity (with acceptable quality) within time. To realize the ANTICO concept, we integrated several AI technologies including probabilistic plan recognition (intent prediction module), Bayesian reasoning (information adapter and information presenter modules), and constraint optimization (information gatherer module). The agent architecture presented in this paper has two additional advantages. First, our approach is domain independent so that various types of applications can be developed from it. As a proof of concept application, we demonstrated the ANTICO concept in the context of emergency response scenario. Second, our modular agent architecture provides flexibility in choosing the main algorithms used in each module without affecting the rest of the system. For instance, we plan to enhance the information adapter module by exploiting meta-data to dynamically adjust the granularity of the data presented to the user, *e.g.*, extracting title, the first sentence, or abstract only. Such enhancements are transparent to other components since the specific algorithms used inside each module are independent.

To demonstrate the practicality of the approach, we presented the design and development of an agent-based system that assists a user in making decisions for time critical tasks in an emergency response scenario. This implementation has been evaluated through simulation, showing that even in situations where the agent's plan recognition and information adaptation fail more than half of the time, the potential performance improvements are significant. As future work, we intend to perform larger scale studies using actual human subjects. To accomplish this, we are integrating ANTICO to a mission management user interface in wide use among the US armed forces, which should allow us to carry out tests with experts in the use of this interface.

Acknowledgement

Omitted for blind review.

9. REFERENCES

[1] J. K. Benini. National planning scenarios, version 21.3. Homeland Security Council, March 2006.

[2] R. Bjarnason, P. Tadepalli, A. Fern, and C. Niedner. Simulation-based optimization of resource placement and emergency response. In *Proc. IAAI*, pages 47–53, 2009.

[3] J. R. Carbonell. A queuing model of many-instrument visual sampling. *IEEE Transactions on Human Factors in Electronics*, 7(4):157–164, 1966.

[4] R. P. Carver. *Reading rate: A review of research and theory.* San Diego, CA, US: Academic Press, 1990.

[5] Foo and Bar. Blind review: proactive agents. 2011.

[6] K. C. Hendy, J. Liao, and P. Milgram. Combining time and intensity effects in assessing operator information-processing load. *Human Factors*, 39(4):30–47, 1997.

[7] Y. Liu, R. Feyen, and O. Tsimhoni. Queueing network-model human processor (qn-mhp): A computational architecture for multitask performance in human-machine systems. *ACM Transactions on Computer-Human Interaction*, 13(1):37–70, March 2006.

[8] A. Pawling, T. Schoenharl, P. Yan, and G. Madey. WIPER: An Emergency Response System. In *Proc. ISCRAM*, 2008.

[9] L. Rabiner. A tutorial on HMM and selected applications in speech recognition. *Proc. of IEEE*, 77(2):257–286, February 1989.

[10] C. D. Wickens. Multiple resources and performance prediction. *Theoretical Issues in Ergonomic Science*, 3(2):159–177, 2002.