TIBOR: A Resource-bounded Information Foraging Agent for Visual Analytics

Dingxiang Liu, Anita Raja and Jayasri Vaidyanath
Department of Software and Information Systems
The University of North Carolina at Charlotte
Charlotte, NC 28223
{dliu, anraja, jvaidyan }@uncc.edu

Abstract

Visual Analytics is the science of applying reasoning and analysis techniques to large, complex real-world data for problem solving using visualizations. Real world knowledge gathering and investigative tasks are very complex because the problem-solving context is constantly evolving, and the data may be incomplete, unreliable and/or conflicting. We describe a mixed-initiative reasoning agent that will assist investigative analysts to choose from and reason about enormous databases of text, imagery, video and webcast. This agent leverages sequential decision making and an AI blackboard system to support hypothesis tracking and validation in a highly uncertain environment. Resource-bounded control mechanisms enable the agent to reason about the uncertainty. We have also designed a user interface that will enable analysts to gather and sift large amounts of evidence and to collaborate with and, where necessary, to control the agent.

1. Introduction

An analyst's problem solving task is complex because the problem context and data are constantly changing. Moreover, the data can be incomplete, unreliable and/or conflicting. This type of problem solving requires reasoning about uncertainty, generation and validation of multiple hypotheses. The incompleteness, unreliability and conflicting nature of the data imply a need for deciding which data sources to query, and what types of analysis to use for collecting, assimilating and abstracting the data into evidence. Moreover the analysis tasks are usually time critical and they have to use approaches ranging from quick and dirty methods to detailed, high quality investigations depending on characteristics the task.

A model of cognitive task analysis performed by analysts has been developed by Pirolli and Card [9]. They have identified two main, overlapping loops in the analyst's problem solving approach, a foraging loop and a sensemaking loop. Figure 1 describes this process. The foraging loop involves accessing external data sources; searching and filtering the data; reading and extracting information and creating the evidence file to (in)validate hypothesis. The sensemaking loop involves iterative generation of hypothesis; presentation of the intermediate and complete hypothesis; and requesting the foraging loop for evidence to support the hypothesis.

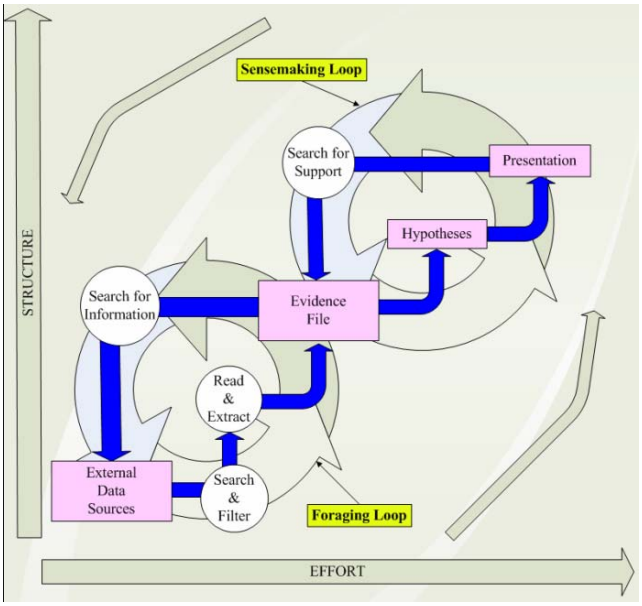

Figure 1: Pirolli and Card's Model of Analyst's Problem Solving Approach

This paper gives an overview of an agent designed to handle the information foraging process, The agent uses an intelligent user interface to assist the analyst in his/her decision making process. Analysis tasks involve identifying and tracking multiple hypotheses by the sensemaking loop. The foraging agent supports the sensemaking loop, by gathering evidence to validate the correct hypotheses and elimination of incorrect hypotheses while solving a query pertaining to Visual Analytics. It uses interactive visualizations [7, 12] to enable analysts to gather and sift large amounts of evidence in a time-bounded fashion and to collaborate with and, where

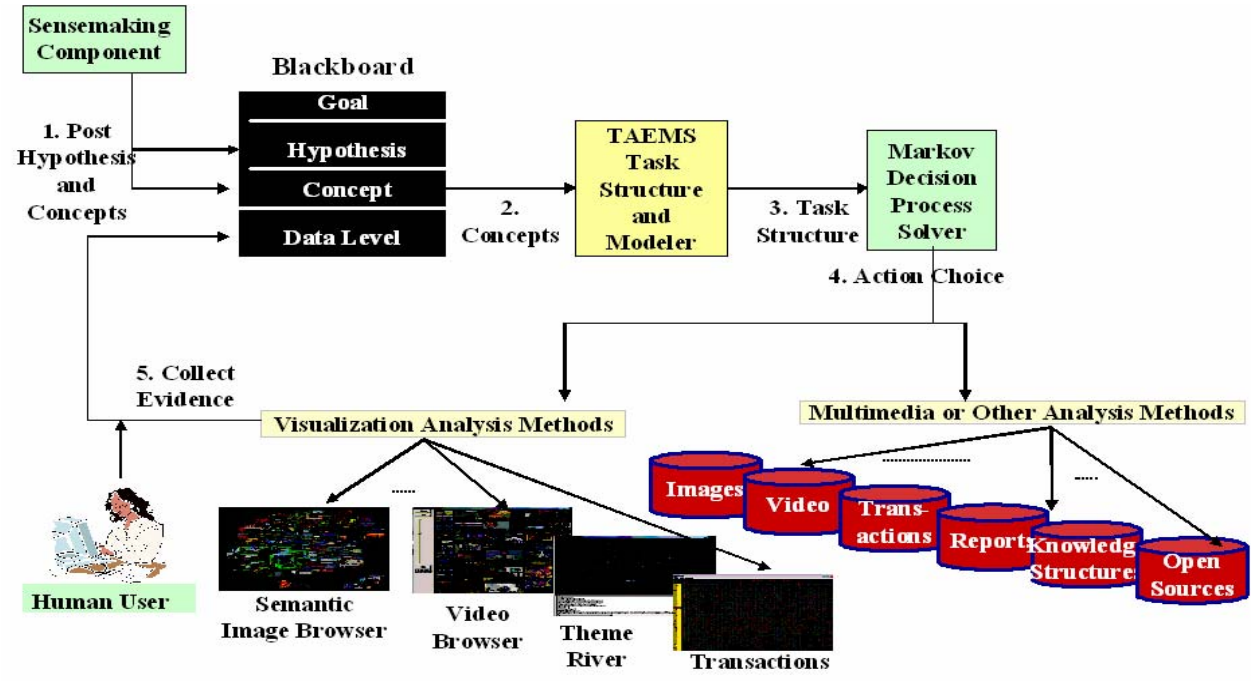

Figure 2: Functional Architecture of Foraging Component

necessary, to control the analysis domain which is inherently dynamic and uncertain. The contributions of our work are (1) a mixed-initiative agent architecture to assist analysts in their foraging tasks (2) using sequential decision making and an AI blackboard approach for gathering evidence to (in)validate hypotheses; and (3) supporting interactive visualization and exploration at every step of the problem solving process.

The rest of this paper is organized as follows. We first describe the blackboard-based information foraging agent and the resource-bounded reasoning techniques used by the agent. We then describe the current status of the system and present empirical results comparing the sequential decision making approach to a deterministic approach. Finally we summarize related work and present our future work and conclusions.

2. A Time-Bounded Information Foraging Agent

We have designed TIBOR, a TIme BOunded Reasoning agent to handle the complex information foraging process required in analysis domains. TIBOR leverages an AI blackboard system [3, 4, 8] and resource- bounded control mechanisms to support hypothesis tracking and validation in a highly uncertain environment like the analysis domain. Figure 2 describes the TIBOR's agent architecture and control flow. TIBOR handles three types of decisions: gathering of large scale, high dimensional data from a variety of sources, which might be linked multimedia data as found on the web, broadcast video, time-dependent transactional data, telecommunication data, or other types of data; determining the type of processing to extract the data from these sources; and determining appropriate interactive visualization of these data. The following is a description of TIBOR's decision making process and control flow. The sensemaking component [1] posts a set of hypotheses and concepts to TIBOR's blackboard (Step 1 in Figure 2) and this triggers the TIBOR agent. The concepts are searchable entities that represent a hypothesis. To support reasoning about time and quality trade-offs, and thus a range of different resource/solution paths, TIBOR contains problem-solving models called Taems [5] task structures. Taems task structures are abstractions of the low-level execution model and capture uncertainty in outcome distributions. They are generated by the Task Structure Modeler sub-component. The Task Structure Modeler takes the concepts as input (Step 2) and generates the corresponding Taems task structure (Step 3) that enumerates multiple different ways (choices for databases, visualization tools and user interactions) to obtain evidence to verify the set of concepts. These different choices are described statistically in the task structure in two dimensions, duration and quality via discrete probability distributions. We provide a detailed description of Taems in the following section. The Taems task structure associated with the concepts is then translated [10, 11] into a Markov Decision Process (MDP) [2] which is also described in detail in the discussion that follows. The

Markov Decision Process Solver computes the optimal policy for the MDP given the resource (deadline) constraints and prescribes the best action (Step 4).

User interactions play an important role in the foraging analysis making this a mixed-initiative agent. TIBOR is equipped with a user interface that will support the mixed-initiative problem solving process. The analyst must be given the ability to manually direct a search or override actions suggested by the control mechanism, in order for this automated agent to be accepted by the analyst community. The contingency plans built into the MDP policy will allow the control system to adjust dynamically to such overrides. The automatic processing of the visualization tools along with the user interactions will determine the confidence in the concepts (Step 5). These evidential data as well their confidence values are then posted on the blackboard. The blackboard will then propagate the confidence values to verify the associated hypothesis. It is crucial for TIBOR to support both opportunistic and planned verification of hypotheses and concepts. It is probable that while gathering information to verify a concept supporting one hypothesis, the belief for a competing hypothesis increases. TIBOR's control component will model this possibility and then opportunistically redefine the problem solving process to gather evidence to verify the competing hypothesis.

The heart of TIBOR agent is the AI blackboard system [4] that has three main components: the Blackboard; Knowledge Sources (KSs); and the Control Component. The blackboard functions as a multileveled database for the information that has been discovered and produced thus far. This information includes raw data, partial solutions and various problem solving states. The levels in TIBOR's blackboard are Goal, Hypothesis, Concept and Data, in order of decreasing granularity. The Goal level stores the resolution information. The Hypothesis level has one or more hypotheses. Each hypothesis contains concepts which are represented in the Concept level. The Data level contains the data/evidences gathered to (in)validate the various hypothesis. The layered hierarchy allows for explicit modeling of concurrent top-down and bottom-up processing, while maintaining a clear evidential path for supporting and contradictory information. The information at a given level is thus derived from the level(s) below it, and it in turn supports the hypothesis at higher levels. For example, when evaluating the validity of a particular hypothesis, the system would examine the reliability of the visualizations used to generate the properties of the object upon which the validation will be made, each of which are in turn different types of visual or text data.

The KSs are independent computational modules that together contain the expertise needed to solve the problem. They vary in their internal representation and computational methods and do not interact directly with each other. In TIBOR agent, the KSs include databases of images, video, text and electronic transactions; the visualization tools used to interact with these databases; and the sensemaking component will serve as KSs. Some examples are ImageDBKS, VideoDBKS, IntelReportsKS, SemanticImageBrowserKS, TextKS, and SenseMakingKS etc. A blackboard directs the problem-solving process by allowing KSs to respond opportunistically to changes on the blackboard and chooses the most appropriate KS activation to execute based on the state of the blackboard and the set of triggered KSs [4]. The control component makes runtime decisions about the problem solving process, specifically for a given hypothesis and resource (time) constraints, it will determine the databases and tools that need to be accessed. We have modeled this control process as a Markov decision process-based [2] sequential decision problem. The essence of sequential decision problems is that decisions that are made in resource-bounded, uncertain environments can have both immediate and long term effects; the best action choice depends on the types of future situations.

3. Taems-based Uncertainty Reasoning

As described earlier, TIBOR receives concepts from the sensemaking component and generates the corresponding Taems task structure [5]. For the analysis domain, the Taems task structure contains the various choices of databases, visualization tools and levels of user interaction relevant to the particular query/hypothesis. The Taems language models problem solving patterns. We can model the fact that one of several actions may be performed and also that a given method may have several possible outcomes, or that an agent has the option to perform a facilitating task before the actual one. A quantitative representation of the agent's choices using Taems allows the agent to select that which is most appropriate in the given context [6].

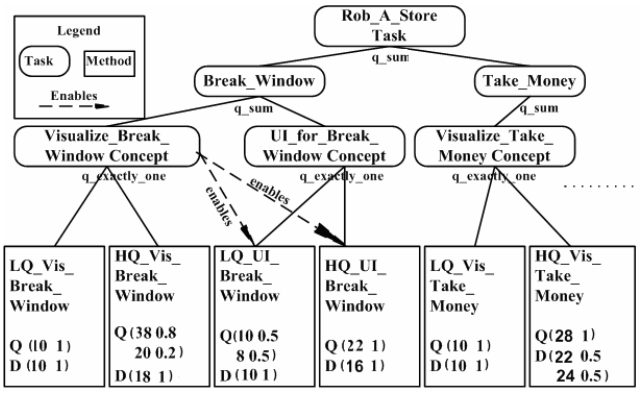

Figure 3: Taems Task Structure for Rob_A_Store Task

Figure 3 describes a simple example Taems task structure that involves finding the evidence for the

hypothesis Rob_A_Store. The top-level task is decomposed into two subtasks, *Break_Window* and *Take_Money*. Since TIBOR is a *mixed-initiative* system, user interaction plays an important part in the problem solving process. Rob_A_Store is modeled in the task structure by the *q_sum* quality attribution factor (qaf). The *q_sum* qaf implies that the quality for the Rob_A_Store task is the sum of the qualities of its *Break_Window* subtask and *Take_Money* subtask. The task *Break_Window* has two subtasks, the first subtask *Visualize_Break_Window* will determine the mode for the semantic image browser tool [12]; and the other subtask *UI_for_Break_Window* will determine the data interaction quality by the user. The qaf associated with the *Break_Window* subtask is also a q_sum meaning its quality is the sum of the *Visualize_Break_Window Concept subtask and* the *UI_for_Break_Window Concept* subtask.. *Visualize_Break_Window* and *UI_for_Break_Window* have two subtasks respectively. A *q_exactly_one* qaf is functionally equivalent to an XOR operator. The quality of the *Visualize_Break_Window Concept* is the quality of any of its subtasks, provided that only one subtask has quality.

Primitive actions in Taems, called *methods*, are characterized statistically in two dimensions: quality and duration. Quality is a deliberately abstract domain-independent concept that describes the contribution of a particular action to overall problem solving. Thus, different applications have different notions of what corresponds to quality. We plan to obtain the quality and duration distributions for the tools and user interaction tasks from statistical studies of tools use and analyst user interaction. A quality distribution Q (38 0.8 20 0.2) implies that the action will obtain a quality of 38, 80% of the time and a quality of 20, 20% of the time. *LQ_Vis_Break_Window* and *HQ_Vis_Break_Window* in Figure 3 are the two alternative ways of visualizing the data related to the *Visualize_Break_Window Concept*. The former will open up the images quickly in low quality mode using a low pixel rate; the latter will take a longer time to load the images but will have higher precision of the images and obtain greater quality. The enables non-local relationship from the *Visualize_Break_Window Concept* to the methods *LQ_UI_Break_Window* and *HQ_UI_Break_Window* implies that the *Visualize_Break_Window Concept* has to accrue non-zero quality for the primitive actions related to user interaction to be successful. In other words, the images in the database have to be successfully rendered by the visualization tool for the user to have useful interactions with the data.

In order to determine the optimal action choices, the Taems task structure is translated into a Markov Decision Process by initializing a state set and identifying the possible actions and expanding each possible outcome which are characterized by discrete quality, cost and duration values.

4. Markov Decision Process

A Markov Decision Process (MDP) [2] is a probabilistic model for decision making and planning. It uses dynamic programming to decide on the optimal policy that yields the highest expected utility. MDPs capture the essence of sequential processes and are used to compute policies that identify, track, and plan to resolve confidence values associated with blackboard objects, which in this application correspond to evidence and hypotheses about the evidence.

A Markov Decision Process has four components: a set of actions (A), a set of states (S), a probability function (P), and a reward function (R). $P^a(ss')$ is a probability function denoted as the probability of transitioning from state s to s' via executing action *a*, while $R^a(S)$ is a reward function defined by the reward received for taking action *a*. The solution to a MDP is a policy π which is a mapping from states to actions. The value function $V^\pi(s)$ is the expected cumulative reward of executing policy π starting in state s. It can be expressed as

$$V^\pi(s) = E\,[r_{t+1} + r_{t+2} \ldots | \, s_t = s, \pi]$$

Where r_t is the reward received at time t, s_t is the state at time t.

An optimal policy is one that maximizes the expected value of the policy. The optimal value function, written as V*, is associated with a specialized form of the Bellman equations:

$$V^*(s) = \max \sum_{s'} P(s' \mid s, a)\,[R(s' \mid s, a) + V^*(s')]$$

The MDP solver receives a task structure along with an associated deadline from the Taems task structure modeler sends the task structure it generates to the MDP solver. This deadline to validate the hypothesis can be specified by either the user or the sensemaking component. Suppose the deadline is provided as an input to the system and a policy

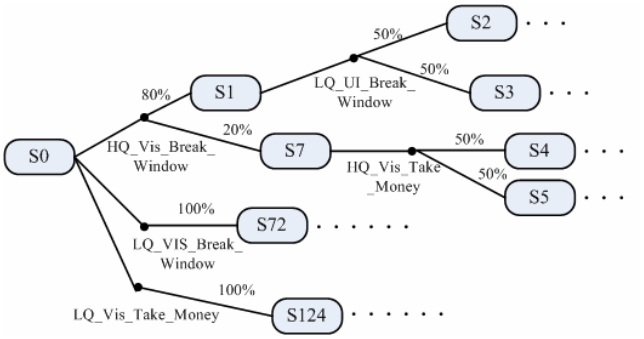

Figure 4: A partial view of the Markov Decision Process describing states, actions and transition probabilities for the Rob_A_Store task

is computed as described in section 2. Figure 4 shows the policy computed for a deadline of 85 time units.

The policy obtained from the MDP assists the analysts in looking at the trade-offs between the greedy choice of actions and the optimal choice of actions in dynamic and uncertain domains.

5. TIBOR In Execution

Our current implementation of TIBOR leverages the Taems representation and the MDP-based sequential decision making process. It also supports the semantic image browser [12] and allows for user interaction. We plan to integrate the more complicated reasoning of blackboard into the agent in the near future. The following is a description of the implemented system along with screen shots when executing the Rob_A_Store task with a deadline of 85 time units.

TIBOR has a control panel, shown in the left window of Figure 5 that allows the user to interact with the agent by specifying his/her choices and also to track the progress of the problem solving process. It can be thought of as a dashboard having all the controls to manage the user's decision-making process. The control panel provides two types of views, the current view provides the decisions made by system on sources, analysis tools and visualization tools as well as interaction decisions on interactions made by the user and the future view provides the analyst choices about data sources, analysis tools, visualization tools and hypothesis. The control panel also has a time slot that keeps the user informed about the time used. The Executed Methods window lists all the actions that have been executed so far.

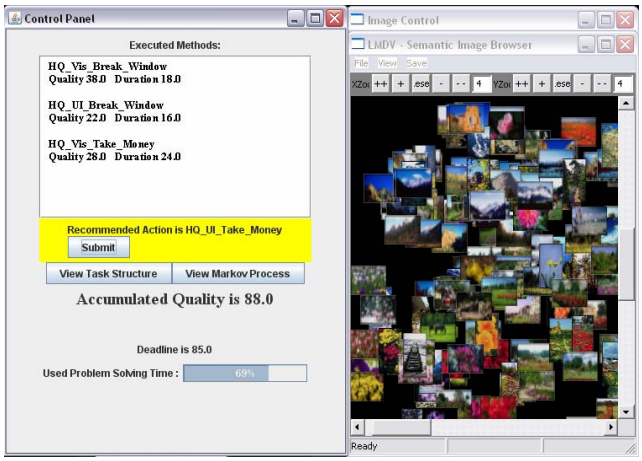

Figure 5: Screenshot of TIBOR in the midst of executing HQ_UI_Take_Money method: the control panel is on the left and the semantic image browser on the right

The system is designed to run in two modes, minimized (precognitive) mode and the maximized mode. The minimized mode is as shown in Figure 5. The maximized mode will display the entire Taems structure under consideration and the corresponding MDP. The View Task Structure button displays the TAEMS task structure as a snapshot of the current state of problem solving. The View Markov Process button will enable the users to view the decision process as well as the optimal policy. The control panel also provides the user with the accumulated quality to show the quality accrued after each action completes execution. When an action is chosen the control panel triggers the associated visualization tool (the semantic image browser) as shown in the right window of Figure 5. The agent continues its execution providing recommendations for user interaction along the way. The left window of Figure 5 is the screen shot of the control panel. It provides a list of actions executed, the corresponding quality and duration values, the accumulated quality for the top-level task and used problem solving time.

Table 1: Comparison of Quality of Deterministic Schedule (DS) and the policy (POL) computed for the corresponding Markov Decision Process. DL is deadline; AQ is average quality and SD is the standard deviation

DL	AQ_{DS}	SD_{DS}	AQ_{POL}	SD_{POL}	t-test
10	0	0	20	2.11	2.4837E-10
28	33.2	6.20	47.6	5.06	0.00095931
32	34.4	5.80	60.6	4.62	2.52176E-09
46	58.8	3.80	88.6	8.28	4.13636E-06
68	85.6	5.06	103.6	5.06	5.5741E-05
85	107.6	5.06	107.6	5.06	1

6. Empirical Results

In this section, we describe our efforts to evaluate the MDP-based decision making mechanism for task structures representing different deadlines. As described earlier, the MDP approach produces a policy that will dynamically adjust the problem solving process to the deadline and runtime execution characteristics. We also define a deterministic process scheduler that builds a static schedule with the highest possible quality for the deadline. An example of deterministic schedule (DS) in Figure 3 is:

{*HQ_Vis_Break_Window*, *HQ_UI_Break_Window*, *HQ_Vis_Take_Money*, *HQ_UI_Take_Money*}. These task structures are variations of structure in Figure 3.

Due to the uncertainty in method execution as described in section 3, there is uncertainty in the quality and duration distributions of the deterministic schedule (DS). For example, although not shown in the Figure 3, the quality distribution for method *HQ_UI_Take_Money* is (20 0.6 10 0.4) and duration distribution for this method is (26 0.4 36 0.6). So the quality distribution for DS of the task structure in Figure 3 is (108 48% 98 32% 90 12% 80 8%) and the

duration distribution is (82 20% 92 30% 84 20% 94 30%) which captures the uncertainty in the schedule execution.

The experiment was designed to compare the quality and duration of the MDP policy for task structure representing different deadlines to that produced by the deterministic plan. We generated six example task structures based on the structures in Figure 3 with deadlines of 10, 28, 32, 46, 68 and 85 respectively. We also varied the quality and duration distributions of the leaf nodes for each example.

Given the duration distributions of the task structures, deadlines in the 60-100 range are considered loose deadlines, 20-60 time units are considered medium tightness deadlines and 0-20 time units are categorized as a tight deadline. We used the MDP-based controller to generate policies and ran ten simulations for each task structure and recorded the quality obtained at the top-level task as well as the total execution time for each run. Then the deterministic schedule was executed ten times and we recorded the average quality obtained for each deadline as well as the average execution time.

The results of the experiments comparing quality are shown in Table 1. Each row in the table represents a specific deadline. AQ_{DS} and SD_{DS} are the quality and standard deviation obtained by averaging results from ten real-time simulation runs of the deterministic schedule computed for each of the six task structures. AQ_{POL} and SD_{POL} are the quality and standard deviation obtained by averaging results from ten real-time simulation runs of the MDP computed for each of the six task structures. Column four represents a two-tailed t-test for qualities by ten times executions of MDP method and deterministic plan. Row one represents the task structure with a tight deadline. For the first five rows, the t-test value is less than 0.05. Hence for these deadlines, performance of MDP schedule is statistically significantly different from the performance of the deterministic plan. For row six, the t-test value is 1. Hence there is no significant difference for deadline 85. Because the deadline 85 is so loose that deterministic plan can complete all methods to obtain same high quality with MDP method.

The results comparing execution-time schedule durations are shown in Figure 6. All data points except one (when the deadline is 85) show the MDP method uses the total allowed time more effectively than the deterministic schedule. It means MDP method can take advantage of time to obtain maximum quality without exceeding deadline.

We now will describe some sample execution runs to elucidate this discussion. Suppose the deadline is 10 time units, the deterministic schedule does not execute because the duration of the very first method/action *HQ_Vis_Break_Window* is greater than 10 time units (c.f. Figure 3). Suppose the deadline is 46, the deterministic schedule executes *HQ_Vis_Break_Window* and *HQ_UI_Break_Window*. The MDP method adapts its policy and uses the allowed time more effectively to execute actions: *HQ_Vis_Break_Window*, *LQ_UI_Break_Window*, *LQ_Vis_Take_Money*, and *LQ_UI_Take_Money*. So for deadline 46, the MDP method produces greater quality than the deterministic method. Suppose the deadline is 85,

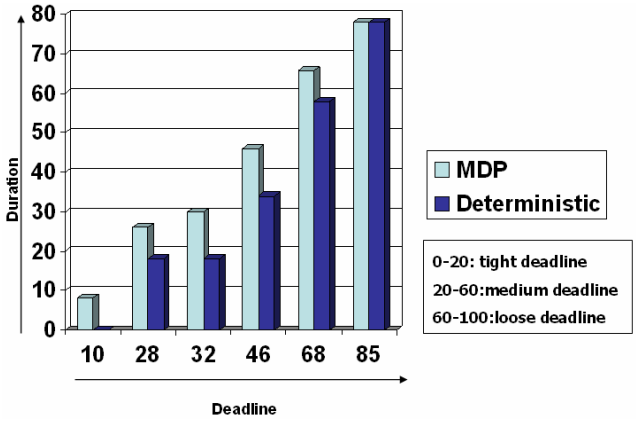

Figure 6: Comparison of Duration

this deadline is so loose that both the MDP and deterministic approach execute the following schedule {*HQ_Vis_Break_Window*, *HQ_UI_Break_Window*, *HQ_Vis_Take_Money*, *HQ_UI_Take_Money*} and produce same quality. Thus both Table 1 and Figure 6 show that the MDP policy performs better than the deterministic method as it is able to adapt to the uncertainty and dynamism of the runtime environment.

7. Discussion

We have described TIBOR, a mixed-initiative agent capable of assisting humans in complex analysis tasks using visualizations and an intelligent user interface. We have identified abstract representations of these tasks to assist in the automated analysis as well as integrated the agent with an image database and the semantic image browser visualization tool. We have also implemented an MDP-based resource-bounded control mechanism that will reason about uncertainty in the environment as well as in the complex real-world data.

The sensemaking KS [1, 9] will be invoked to perform identification of novel situation story understanding and plan recognition. Additionally, in novel situations, the sensemaking KS should be equipped to handle situation construction, new hypothesis generation, and plan adaptation. TIBOR's foraging capabilities, on the other hand, include identifying interesting data to initiate the problem solving process as well as efficiently gather evidence to support/refute hypothesis generated by the sensemaking KS.

Blackboard-based architectures have been previously used for complex information gathering and analysis tasks. Morrison and Cohen [3] use a Bayesian blackboard called

AIID to serve as a prototype system for analysis and data fusion. BIG (Lesser, V. et al., 2000) is a resource-Bounded Information Gathering agent that combines several sophisticated AI components. BIG gathers web-based information, extracts information from both unstructured and structured documents, and reaches a decision. BIG uses an opportunistic linear planner and scheduler to direct its activities. TIBOR focuses on the end-to-end decision reasoning of analytical tasks and uses a MDP-based approach to reason about the inherent uncertainty of the domain. TIBOR is designed to be a fully-functional mixed-initiative agent that leverages the state-of-the-art in visualization tools. The control panel in TIBOR also enables the human user to track the problem solving process at various levels of abstraction.

Our next step is to complete integration of the AI blackboard and a variety of knowledge sources, including a sensemaking knowledge source that uses case-based reasoning and pattern recognition; as well as other visualization tools such as the semantic video browser [7]. Our goal is to ensure that TIBOR scales to real-world for example, investigating analysis problems, determining if there will be a pandemic in a certain region in the next two months, or if a certain person is involved in illegal activities. This includes equipping the blackboard to reason about multiple complex hypothesis simultaneously.

8. Acknowledgement

We thank William Ribarsky and Robert Kosara for discussions on TIBOR's interface. We also thank Kristin Cook and the anonymous reviewers for their feedback. We would like to thank Ashok Goel and his team at Georgia Tech for sharing the details of the Rob_A_Store pattern which we used to generate the corresponding TAEMS task structure. These patterns were obtained as a result of processing the VAST-2006 data set (http://conference.computer.org/vast/vast2006). This work was sponsored by the National Visualization and Analytics Center (NVAC) under the auspices of the Southeastern Regional Visualization and Analytics Center. NVAC is a U.S. Department of Homeland Security Program led by Pacific Northwest National Laboratory.

References

[1] Adams, S. and Goel, A., *A STAB at Making Sense of VAST Data.* To Appear in AAAI-2007 Workshop on Plan, Activity, and Intent Recognition, Vancouver, Canada, July 23, 2007.

[2] Bertesekas, D. and Tsitsiklis, J., *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA., 2006.

[3] Morrison, C. and Cohen, P., *The Hats Simulator and Colab: An Integrated Information Fusion Challenge Problem and Collaborative Analysis Environment*. ISI 2006: 105-116.

[4] Corkill, D., *Blackboard Systems* AI Expert, 1991, 6(9): pp. 40-47.

[5] Decker, K., *TAEMS: A framework for environment centered analysis and design of coordination mechanisms*, in Foundations of Distributed Artificial Intelligence, Chapter 16, G O'Hare and N. Jennings (eds.), Wiley Inter-Science, January 1996, pp. 429-448.

[6] Horling, B., Lesser, V., Vincent, R., Wagner, T., Raja, A., Zhang, S., Decker, K., and Garvey, A., *The TAEMS White Paper* Unpublished, January 1999.

[7] Luo, H., Fan, J., Yang, J., Ribarsky, W., and Satoh, S., *Exploring Large-Scale Video News via Interactive Visualization*. Charlotte Visualization Center Technical Report, CVC-UNCC-06-04, 2006.

[8] Lesser, V., Horling, B., Klassner, F., Raja, A., Wagner, T., and Zhang, S., *BIG: An Agent for Resource-Bounded Information Gathering and Decision Making* In Journal of Artificial Intelligence, 2000 Special Issue entitled 'Internet Applications', May 2000, vol 118, issue 1-2, pp. 197-244.

[9] Pirolli, P. and Card, S., *The Sensemaking Process and Leverage Points for Analyst Technology as Identified Through Cognitive Task Analysis*. Proceedings of 2005 International Conference on Intelligence Analysis, 2005, pp. 2-4.

[10] Raja, A., Lesser, V. and Wagner, T., *Toward Robust Agent Control in Open Environments*. Proceedings of the Fourth International Conference on Autonomous Agents, ACM Press, 2000,pp.84-91.

[11] Raja, A. and Lesser, V., *A Framework for Meta-level Control in Multi-Agent Systems*. Proceedings of Autonomous Agents and Multi-Agent Systems, 2007.

[12] Yang, J., Fan, J., Hubball, D., Gao, Y., Luo H. and Ribarsky, W., *Semantic Image Browser: Bridging Information Visualization with Automated Intelligent Image Analysis*. Charlotte Visualization Center Technical Report CVC-UNCC-06-02, 2006.