Demonstration of Teamwork in Uncertain Domains using Hybrid BDI-POMDP Systems

Tapana Gupta*, Pradeep Varakantham*, Timothy W. Rauenbusch+ , Milind Tambe*
* University of Southern California, Los Angeles, CA 90089, {tapanagu, varakant, tambe}@usc.edu
+ SRI International, Menlo Park, CA 94025, rauenbusch@ai.sri.com

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence - Multi-agent Systems

General Terms

Experimentation, Human Factors

Keywords

Multi-agent systems, Personal Assistants, Teamwork, Hybrids

1. OVERVIEW

Personal Assistant agents are becoming increasingly important in a variety of application domains in offices, at home, for medical care and many others [5, 1]. These agents are required to constantly monitor their environment (including the state of their users), and make periodic decisions based on their monitoring. For example, in an office environment, agents may need to monitor the location of their user in order to ascertain whether the user would be able to make it on time to a meeting [5]. Or, they may be required to monitor the progress of a user on a particular assignment and decide whether or not the user would be able to meet the deadline for completing the assignment. Teamwork between such agents is important in Personal Assistant applications to enable agents working together to achieve a common goal (such as finishing a project on time). This working demonstration shows a hybrid(BDI-POMDP) approach to accomplish such teamwork.

Agents must be able to make decisions despite observational uncertainty in the environment. For example, if the user is busy and does not respond to a request from its personal assistant agent, the agent loses track of the user's progress and hence, cannot determine it with certainty. Also, an incorrect action on the agent's part can have undesirable consequences. For example, an agent might reallocate a task again and again even if there is sufficient progress on the task. In the past, teamwork among Personal Assistant agents typically has not addressed such observational uncertainty. Markov Decision Processes [5] have been used to model the agent's environment, with simplifying assumptions regarding either observational uncertainty in the environment or the agent's observational abilities.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
AAMAS'07 May 14–18 2007, Honolulu, Hawai'i, USA.
Copyright 2007 IFAAMAS .

Partially Observable Markov Decision Processes (POMDPs) are equipped to deal with the inherent uncertainty in Personal Assistant domains. Computational complexity has been a major hurdle in deploying POMDPs in real-world application domains, but the emergence of new exact and approximate techniques [8] recently shows much promise in being able to compute a POMDP policy for an agent in real time. In this demonstration, we actually deploy POMDPs to compute the Adjustable Autonomy policy for an agent based on which the agent makes decisions.

Integrating such POMDPs with architectures that enble teamwork among personal assistants is then the next key part of our demonstration. Several teamwork models have been developed over the past few years to handle communication and coordination between agents [7]. Machinetta [6] is a proxy-based integration architecture for coordinating teams of heterogeneous entities (e.g. robots, agents, persons), which builds on the STEAM teamwork model. Machinetta is designed to meet key challenges such as effective utilization of diverse capabilities of group members, improving coordination between agents by overcoming challenges posed by the environment and reacting to changes in the environment in a flexible manner. We use Machinetta proxies to co-ordinate the agents in our demonstration.

Machinetta enables integrating POMDPs and also enables interfacing with BDI architectures that may provide us team plans. In particular, we interface with the SPARK agent framework [2] being developed at the Artificial Intelligence Center of SRI international. SPARK is a Belief-Desire-Intention (BDI) style agent framework grounded in a model of procedural reasoning. This architecture allows the development of active systems that interact with a constantly changing and unpredictable world.

By using BDI-based approaches for generating team plans for agents as well as communication and coordination, and POMDPs for adjustable autonomy decision making, we arrive at a hybrid model for multiagent teamwork [3] in Personal Assistant applications. The following sections describe the application domain in which we deploy this hybrid system as well as the interaction between various components of the system, and its working.

2. MOTIVATING DOMAIN: HIRING SCENARIO

The application domain is a stylized hiring scenario. The team is made up of three users: Karen, Ray and Ken, and their respective agents. Figure 1 shows the structure of this team. The team is responsible for performing the high-level task of hiring a new employee. This task decomposes into subtasks which themselves decompose into further subtasks as shown in Figure 2.

We call the highest-level task a *level 0 task*. A level 0 task decomposes into level 1 tasks which decompose into level 2 tasks and

978-81-904262-7-5 (RPS) ©2007 IFAAMAS

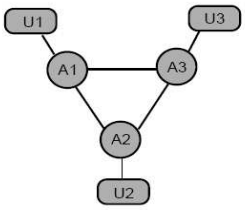

Figure 1: Three personal assistants, A1, A2 and A3 are seen to help teamwork among users U1, U2 and U3

so on. We assume a unique decomposition for each task, and that level 2 tasks are not decomposable. In our hiring scenario, there are three level 1 tasks, and 9 level two tasks.

There are five discrete levels to indicate task progress on level 2 tasks: 0%, 20%, 40%, 60%, 80% and 100%, with 100% indicating that a level 2 task is complete. Level 1 tasks are complete when each of its constituent level 2 tasks are complete. Time is discretized into time steps, with task progress computed at each time step.

We assume that each team member is capable of performing any and all of the level 2 tasks, but one agent may be able to complete a task more efficiently than another. A transition function encodes the efficiency of task completion for each agent and each task. For instance, Ray's performance on a particular level 2 task on which no progress has been made may be encoded as $[0\% \rightarrow 20\%(0.5), 0\% \rightarrow 100\%(0.5)]$. This means that if at time step t Ray's progress on the task is 0%, then at time step $t+1$, his progress will be 20% with probability 0.5, and 100% with probability 0.5.

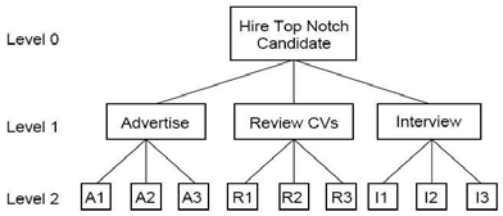

Figure 2: Team plan executed by the personal assistants

The goal of our system is to facilitate the assignment of tasks to agents in order to complete the level 0 task as quickly as possible. The main function of the system is to allocate and reallocate tasks based on an agent's task transition function and the current progress on a task. It is assumed that task progress is monotonically increasing and remains with a task. That is, if at time step t progress level of p for a task has been achieved by Ray, the progress level for that task at time steps greater than t will be equal to or greater than p, regardless of whether that task has been allocated to Karen or Ken.

3. SYSTEM DESCRIPTION

Each agent contains a Machinetta proxy that is responsible for reasoning about and communicating allocation decisions. SPARK sends a team plan to Machinetta with Agent names, their workloads, progress and the tasks that need to be allocated to these agents. Machinetta assigns tasks to each agent using its role allocation algorithm which uses a DCOP, also used for role reallocation [6, 4].

Figure 2 shows the interactions between the various components

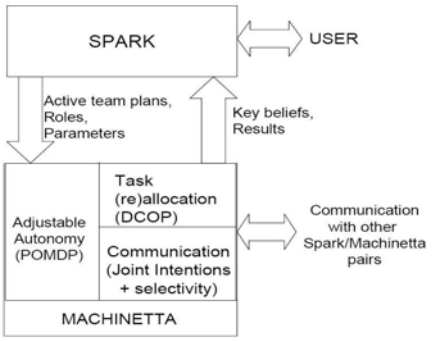

Figure 3: Architecture and Interface of a single Personal Assistant

of the hybrid system. When a task is assigned to an agent, the Adjustable Autonomy (AA) component of Machinetta computes the AA policy for that particular task using a POMDP. The policy tells Machinetta what action to take in response to certain observations about the agent's progress. These actions include doing nothing (wait), ask the user or reallocate the task. SPARK sends beliefs to Machinetta at certain time intervals, which contain the percentage of progress accomplished for a particular task. The AA reasoning uses these beliefs as its observations for making AA related decisions. Machinetta can query the user (via SPARK) for a response or a decision regarding reallocation of a task, in response to which SPARK sends an appropriate response based on its beliefs (updated by user input). If the decision is made to reallocate a task, Machinetta reallocates the task.

The SPARK BDI team plan, which contains information about an agent's workload and progress serve as input for the POMDP to compute the AA policy for a task executed by that agent. The beliefs of the agent serve as observations for the POMDP policy on the basis of which AA decisions are made by Machinetta. The POMDP policy, in turn alters the beliefs of the agents, for example, by reallocation. Thus, by integration of SPARK and Machinetta, we have achieved a hybrid of BDI and POMDP frameworks working synergistically in a multi-agent system.

4. REFERENCES

[1] http://www.ai.sri.com/project/calo, http://calo.sri.com, 2003.
[2] D. Morley and K. Myers. The spark agent framework. In *AAMAS*, 2004.
[3] R. Nair and M. Tambe. Hybrid bdi-pomdp framework for multiagent teaming. *JAIR*, 23:367–420, 2004.
[4] P. Scerri, J. A. Giampapa, and K. P. Sycara. Techniques and directions for building very large agent teams. In *KIMAS*, 2005.
[5] P. Scerri, D. Pynadath, and M. Tambe. Towards adjustable autonomy for the real world. *JAIR*, 17:171–228, 2002.
[6] N. Schurr, S. Okamoto, R. T. Maheswaran, P. Scerri, and M. Tambe. *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*. Cambridge University Press, 2004.
[7] M. Tambe. Towards flexible teamwork. *JAIR*, 7:83–124, 1997.
[8] P. Varakantham, R. T. Maheswaran, T. Gupta, and M. Tambe. Towards efficient computation of error bounded solutions in pomdps: Expected value approximation and dynamic disjunctive beliefs. In *IJCAI*, 2007.