Massive Deliberation

William H. Turkett Jr., John R. Rose, and Michael N. Huhns • *University of South Carolina*

Agents are proliferating on the Web, making it conceivable that their collective reasoning ability might someday be harnessed for robust decision-making. Projects such as SETI@home (http://setiathome.ssl.berkeley.edu/), which processes radio telescope signals, and Folding@Home (http://folding.stanford.edu/), which analyzes protein folding, have demonstrated that massive computational power can be brought to bear on well-defined problems involving large amounts of data and known solution algorithms. The hope is that massive *deliberation power* can soon help solve problems that require knowledge, reasoning, and intelligence.

Until recently, working individually or in small groups, agents across the Web could barely communicate and could only reason under conditions of severely bounded rationality. Projects such as Agentcities (http://www.agentcities.org) showed that widespread heterogeneous agents could collaborate on specific predefined tasks and provide diverse agent-based services. When the tasks are dynamic, of long duration, and ill defined, however, success requires planning that is continual, distributed, and accounts for the social fabric into which the plans and their execution must fit.

Distributed Planning

The systems employed in future domains and their environments will be much larger than those used today. They might include hundreds or thousands of distinct agents that have unique preferences on how to perform multiple tasks. Because of these preferences, and because agents might have difficulty accurately sensing the environments, any given task can have several possible outcomes. Domain environments will also evolve rapidly, at rates of change similar to those seen in everyday human activity.

Given these assumptions, several natural consequences exist. The state space for a planning agent represents all possible views of the world that an agent might have. In complex domains, this state space will be extremely large, hindering the use of centralized planning, and highly dynamic, preventing development of good contingency plans offline. There is also a high probability of agents encountering situations in the environment that cannot be modeled beforehand. An agent's reasoning approach should be capable of handling uncertain and conflicting information and adapting at a rate similar to the environment's rate of change. Finally, because actions might have multiple outcomes, planning algorithms must account for each possible outcome and its likelihood of occurrence.

By posing a planning problem as a *Markov decision problem* (MDP), a designer can develop an optimal policy over domain states, denoting optimal actions to be performed in each state.[1] With the size of the domains, number of agents, and number of actions that are proposed for future problems, however, the standard MDP approach does not scale as desired. We need an alternative approach that can support planning in these types of environments. One such approach is to use localized decision-theoretic, continual planners.[2] Localized planning distributes the search space to individual agents, decision theory lets agents reason over uncertain actions and states, and continual planning lets agents adapt rapidly to changes in the environment and global goals.

Societal Agents

Humans, unlike current agent systems, are extremely good at solving distributed planning problems. Consider how smoothly traffic flows on a normal workday, even though no central plan exists to tell commuters how or where to drive. One reason humans can plan and execute so well is that we share values in the form of ethics and norms that have evolved to maintain social order.

Agent societies might maintain order and focus through analogous ethics and norms, but they need a way of mapping these abstract societal values to utilities that can be expressed in a computational framework that an agent can use.

Human societal laws are expressed as abstract concepts and qualitative statements. Accurately expressing such concepts in a computational framework requires a methodology for mapping shared abstract principles to preferences over world states and utilities over actions, as shown in Figure 1. Agents can then use decision theory in their negotiations to evaluate the expected utility of proposed actions and resource usage. A fundamental concept underlying planning objectives is relative preference over the possible outcomes of a plan. These preferences let an agent rationally choose the plan that it believes best achieves its goals. Utility theory provides a framework for representing preferences and reasoning quantitatively over such preferences, but it will also be desirable in certain problem domains to assess the preferences qualitatively. Integrating distributed planning with the social values concept described earlier will enable dynamic, rational, and massively distributed planning and plan execution at multiple levels of granularity.[3]

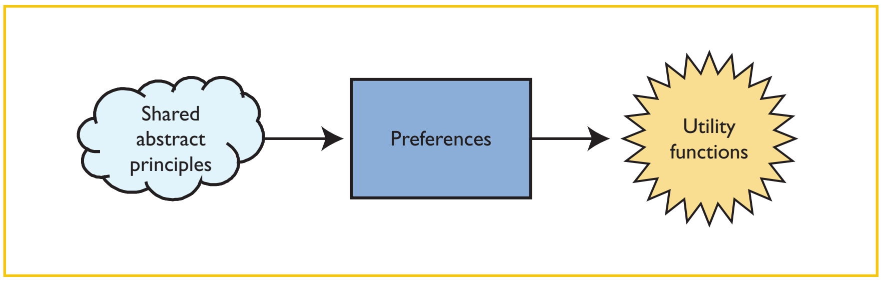

Figure 1. Mapping from shared abstract principles to utility functions. To implement shared ethics and norms, agents map abstract social principles to personal preferences, which are then used to define action utilties during plan generation.

Planning Framework

At the University of South Carolina, we are developing a cooperative distributed initiative framework that allows global coordination to emerge from the interaction of agents that plan using local knowledge.[4] This distributed cooperative framework, Eplan,

- distributes the planning search space to individual agents through localized planning,
- lets agents use decision theory to reason about uncertain actions and states,[1] and
- enables continual planning that lets agents adapt rapidly to changes in environment and objectives.[2]

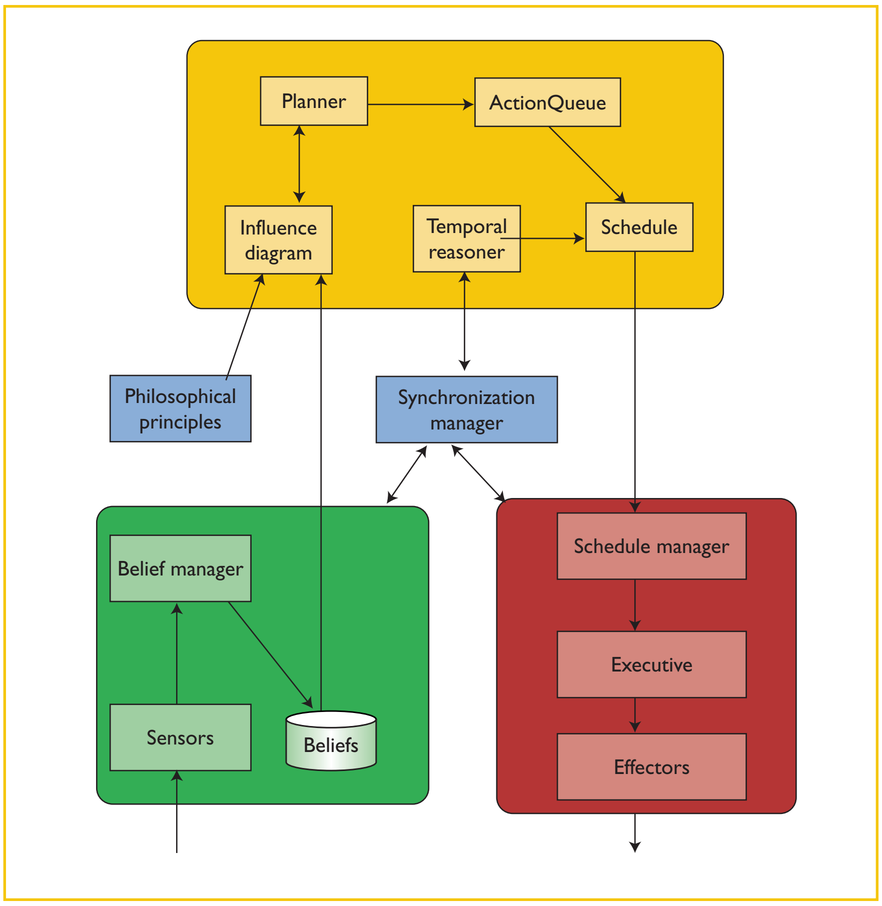

Figure 2. Local planner architecture. A local planning agent's minimum architecture includes four components in the EPlan framework.

The framework is implemented through a local planning architecture in each agent, guidance in planning from shared abstract principles, and multiagent interactions from which distributed planning emerges. Figure 2 depicts a local planning agent's minimum architecture, which we divide into four components: a belief system, a planning system, an execution system, and a synchronization manager.

Belief System

The belief system maintains the agent's current view of the world. Each agent maintains beliefs over the states of its *resources* and *effectors*. A resource, such as fuel or battery power, is consumed during execution, while an effector, such as the wheels or gripper of a robot, actually performs actions. The agent also includes sensors that return knowledge it might require about its environment. Because the sensors might provide ambiguous or inaccurate information, sensory information passes through a belief management system before the agent stores it as beliefs. The belief management system, which is domain specific, maps sensory inputs to a probability distribution over states representing the agent's beliefs.

Planning System

The planning system consists of a dynamic decision network, a planner control mechanism, a queue of actions to be performed, a temporal reasoner for predicting action durations, and a schedule in which actions are mapped to specific execution start times.

The dynamic decision network is the foundation of the planning system because it lets an agent reason about actions. A dynamic decision network (DDN) is a graphical data structure that models the state of the world over time and typically represents a small number of connected time slices. In each time slice, a set of variable nodes represents the state of the world in terms of probability distributions, a decision node represents the actions available to an agent, and a utility node defines expected utilities over the possible states of the world. The agent's current beliefs about the world are set as evidence in the first time slice of a DDN. This is accomplished by setting the probability distribution of the relevant variable nodes such that the state that corresponds to the agent's belief has probability one and all other states have probability zero.

Given this information, the expected utility of performing each possible action can be calculated and the agent can reason over which activity has the highest expected utility. This reasoning is based on both domain knowledge and the abstract social principles to which the agent adheres.

Once the agent selects this action, the dynamic decision network can predict the effects of its execution on the world. By iteratively propagating predicted world states into the next time slice of the decision network, the agent can generate multiple steps in a plan.

Our architecture's planner drives iterations of the dynamic decision network. The planner obtains the agent's beliefs to serve as initial evidence, stores each action the decision network generates in the action queue, and then executes the next dynamic decision network iteration. The planner process can be triggered in four different ways:

- initial planning when the agent is first instantiated,
- replanning if the agent is nearing execution of the complete set of actions generated for the current horizon,
- replanning if beliefs change, or
- replanning if the execution of a current task is taking longer than expected and conflicts with another scheduled action's start.

The *n* steps generated from the dynamic decision network are stored in order in an action queue. After all *n* actions have been generated, the agent uses a domain-specific temporal reasoner to estimate the duration of each action, and then schedules them appropriately.

Execution System

The execution system consists of a schedule manager, an executive, and domain-specific effectors. The schedule manager responds to the agent's clock and transfers actions to the executive when the scheduled time for an action arrives. Depending on the action to be performed, the executive then initiates the domain-specific low-level actions required for execution.

Synchronization Manager

Our framework isolates the four core tasks in the planning process — updating beliefs, removing a task from the schedule to start execution, signaling task completion, and executing the actual planner — through the ordering of their execution. The synchronization manager allows the tasks to be performed in the order it receives them, but ensures that only one planning task is executed at a time. This guarantees consistency in the planning process and prevents planning tasks from interfering with each other. Without synchronization, the schedule manager might spawn actions from an incomplete schedule, or the planner might include partially updated beliefs as evidence in the dynamic decision network.

Cooperative Interaction

The architectural infrastructure that supports collaborative planning and execution has three core stages: coalition formation, coalition plan development, and commitment management. Coalition formation is spawned when an agent cannot perform a required task, perhaps because it does not have the required capabilities or cannot access some required resource. The formation is similar to many other collaboration-formation protocols (such as the contract net[5]) and requires discovery, proposal, and reply. In the discovery phase, a requesting agent searches for an appropriate task agent to perform a required task. We can implement discovery through several mechanisms, including broadcast messages, a directory service, or direct knowledge of an agent's capabilities.

After locating a task agent, the requesting agent sends a proposal that includes necessary constraints for successfully completing the proposed task. The task agent reviews the proposed task plan and constraints and evaluates the utility and its ability to schedule the task's performance. If the task agent believes executing the task is beneficial, it replies with an acceptance that marks entry to the coalition plan development stage. Otherwise, the task agent sends a rejection, causing the requesting agent to search for other task agents or to reexamine its own plan.

During coalition plan development, agents negotiate to integrate their subplans and avoid conflicts. The integration of subplans lets agents work around potentially damaging interactions between their actions. Agents then finalize coalition plans through decision-theoretic commitment management coupled with social principles of commitment. Commitment management is included in decision theoretic planning by assigning extra costs to actions that keep the agent from fulfilling a commitment.

Under this planning framework, each agent's goal is to generate small dynamic plans that fit individual needs in a larger emergent plan, rather than build components of an explicit overall plan. A global approach to reaching global goals emerges from local interactions among agents. This global plan, however, is never required to be explicitly defined and known to the agents, allowing this approach to scale to very large and complex problems. Figure 3 depicts how a plan develops through agents' local interactions. The use of decision-theoretic maximum utility planning, together with shared principles, helps lead agents to select actions that support the emergence of coherent and coordinated interactions, while implicitly acting as filters to prevent actions that are least likely to help achieve global and individual goals or are harmful to the society.

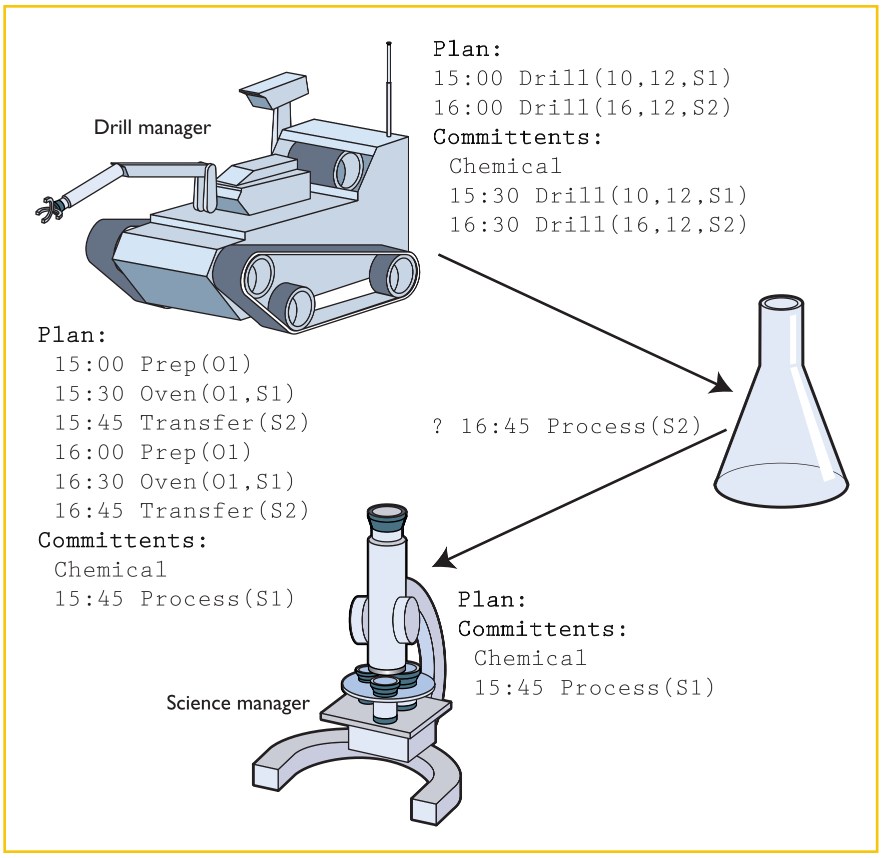

Figure 3. Emergent global planning through local interactions. Science agents develop an implicit global plan while interacting with task requests and commitments during local planning.

Conclusion

This distributed planning methodology provides a step toward a future in which numerous agents can robustly and effectively work together to solve large problems. The potential applications of such an architecture are widespread, both on and off the Internet. The future of the "cognitive Web" and its services promises to open up never-before-seen support for collaborative reasoning and decision-making. Large numbers of agents could interact to solve difficult reasoning problems and agent assistants could participate in agent communities to efficiently and robustly accomplish user goals. Principled agents enforcing ethics in sensitive domains could drive e-government. Off the Web, applications of this framework could extend from the practical realization of coordinated Mars rovers to the creative task of developing coordinated and competitive gaming agents.

Acknowledgments

The U.S. National Science Foundation supported this work under grant number IIS-0083362.

References

1. C. Boutilier, T. Dean, and S. Hanks, "Decision-Theoretic Planning: Structural Assumptions and Computational Leverage," *J. Artificial Intelligence Research*, vol. 11, 1999, pp. 1–94.
2. M. des Jardins et al., "A Survey of Research in Distributed Continual Planning," *AI Magazine*, vol. 20, no. 4, 1999, pp. 13–22.
3. J.R. Rose and M.N. Huhns, "Philosophical Agents," *IEEE Internet Computing*, vol. 5, May/June 2001, pp. 104–106.
4. W.H. Turkett Jr. and J.R. Rose, "An Architecture for Robust Distributed, Decision Theoretic Planners," submitted to *13th Int'l Conf. Automated Planning and Scheduling* (ICAPS-03), 2002.
5. R.G. Smith, "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver," *IEEE Trans. Computers*, vol 13, no. 29, 1980, pp. 1104-1113.

William H. Turkett Jr. is a Ph.D. candidate in computer science and engineering at the University of South Carolina, where he is conducting research on agent-based distributed planning. Contact him at turkett@cse.sc.edu.

John R. Rose is an associate professor of computer science and engineering at the University of South Carolina, where he divides his time between research in the areas of multiagent systems and bioinformatics and innovative computer science education. Contact him at rose@cse.sc.edu.

Michael N. Huhns is a professor of computer science and engineering at the University of South Carolina, where he also directs the Center for Information Technology. Contact him at huhns@sc.edu.