Coordinating Agents' Meta-Level Control

Anita Raja
Department of Software and Information Systems
The University of North Carolina at Charlotte
Charlotte, NC 28223
anraja@uncc.edu

Victor Lesser
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
lesser@cs.umass.edu

Abstract

Embedded systems consisting of collaborating agents capable of interacting with their environment are becoming ubiquitous. It is crucial for these systems to be able to adapt to the dynamic and uncertain characteristics of an open environment. The question of when this adaptation process should be done and how much effort should be invested in the adaptation becomes especially challenging in the context of multi-agent systems. We present a generalized agent framework for meta-level control called GeMEC. We describe GEMEC's decentralized Markov Decision Process (DEC-MDP)-based model for decision making in Netrads, a tornado tracking application. This model will capture interactions where meta-level decisions made in one agent's MDP can affect meta-level MDPs of other agents. The cost of meta-level control can be controlled by constructing and evaluating the DEC-MDPs offline.

Introduction

Embedded systems consisting of collaborating agents capable of interacting with their environment are becoming ubiquitous. It is crucial for these systems to be able to adapt to the dynamic and uncertain characteristics of an open environment. The adaptation needs to be based on the priority of tasks; availability of resources; and availability of alternative ways of satisfying these tasks as well as tasks expected in the future. Some important questions are when this adaptation process should be done; how much effort should be invested in the adaptation as opposed to just continuing with the current action plan; and the ramifications of making these decisions in a multi-agent context.

Bounded rationality (Russell and Wefald 1989), the ability to reason about resource allocation to computation at any point in time, has been used in the context of beliefs, intentions and learning (Doyle 1983), intelligent system design (Horvitz 1988), problem solving and search (Simon and Kadane 1974) and planning (Stefik 1981). Russell, et al. (Russell, Subramanian, and Parr 1993) cast the problem of creating resource-bounded rational agents as a search for the best program that an agent can execute. In searching the space of programs, the agents, called bounded-optimal agents, can be optimal for a given class of programs or they can approach optimal performance with learning, given a limited class of possible programs. Schut and Wooldridge (Schut and Wooldridge 2001) observed that a Markov Decision Process (MDP)-based model toward decision making is most similar to the bounded optimality model. Cox (Cox 2005) provides a review of metacognition research in the fields of artificial intelligence and cognitive science. To our knowledge, meta-level control in complex agent-based settings was first explored in our previous work (Raja 2003; Raja and Lesser 2007), where we developed a sophisticated architecture that could reason about alternative methods for computation, including computations that handled simple negotiation between two agents. This paper builds on results from this earlier work and opens a new vein of inquiry by addressing issues of scalability, partial information and complex interactions across agent boundaries in real domains. It includes defining a generalized agent framework for meta-level control called GeMEC (see Figure 2) . A decentralized Markov Decision Process (DEC-MDP)-based model with the ability to model interactions so that meta-level decisions made in one agent's MDP can affect meta-level MDPs of other agents is proposed.

The generalized framework for meta-level control could significantly impact multi-agent systems research in that one can identify domains and scenarios where meta-level control would be advantageous to overall multi-agent performance. It could also impact other fields that deal with uncertainty and non-stationarity including robotic path-planning. This paper is structured as follows: We first describe the taxonomy of agent decisions and their interdependencies first from a single-agent perspective and then from a multi-agent perspective. The relevance of these research issues with in the context of Netrads, a real-world tornado-tracking application is presented followed by a description of a generalized framework for meta-level control. We then present the conclusions and future work directions.

Taxonomy of Agent Decisions

Agents in embedded systems operate in an iterative three-step closed loop: receiving sensations from the environment; performing internal computations on the data; and responding by performing actions that affect the environment either

using effectors (Sutton and Barto 1998) or via communication with other agents. Two levels of control are associated with this sense, interpretation and response loop: deliberative and meta-level control. This model is comparable to Figure 1 described in (Cox and Raja 2007). The lower control level is deliberative control (also called object level), which involves the agent making decisions about what local problem solving to perform in the current context (also called domain actions) and how to coordinate with other agents to complete tasks requiring joint effort. These deliberations may have to be done in the face of limited resources, uncertainty about action outcomes and in real-time. Tasks in these environments can be generated at any time by the environment or other agents and generally have deadlines where completion after the deadline could lead to lower or no utility.

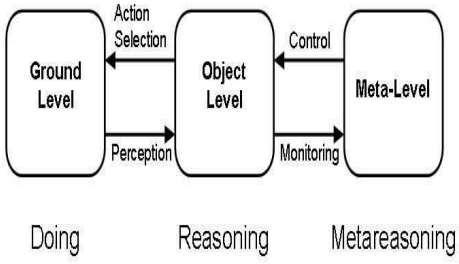

Figure 1: Decision Taxonomy

Single-Agent Meta-Level Control

At the higher control level is meta-level control, which involves the agent making decisions about whether to deliberate, how many resources to dedicate to this deliberation and what specific deliberative control to perform in the current context. In practice, meta-level control can be viewed as the process of deciding how to interleave domain and deliberative control actions such that tasks are achieved within their deadlines and also allocating required amount of processor and other resources to these actions at the appropriate times. For example, suppose the current time is 10 and an agent is in the midst of executing a set of high quality tasks with a deadline to complete the task at time 25. At time 15 the agent receives a new medium quality task T_{new} with expected duration of 10 and a deadline of 40. The sensible meta control decision would be for the agent to delay deliberating about how to accomplish task T_{new} in the context of ongoing activities until the existing task set has completed execution (time 25). This would guarantee that the existing task set completes within its deadline and quality can still be gained by processing T_{new} by time 40. The meta-level control decision process should be designed to be computationally inexpensive, thus obviating the need for meta-meta-level control.

Meta-level control also involves choosing among alternative deliberative action sequences including choosing among various alternatives for scheduling/planning ; choosing between scheduling/planning and coordination; and, allocating extra time for learning activities, etc. Consider the following example: suppose the current time is 6 and an agent has two tasks: T_x, a high quality task with expected duration of 10 and deadline 30; and T_y, a low quality task with expected duration 6 and deadline 30. The meta-control decision could be to spend 5 time units doing a detailed high-quality deliberation about T_x to find a good plan for the high quality task and two units doing a quick and dirty deliberation to generate a plan for T_y (the lower quality task). The remaining time not used by deliberative activities will be allocated to successfully execute both tasks.

Multi-Agent Meta-Level Control

In the multi-agent context, meta-level control decisions at different agents need to be coordinated (Alexander et al. 2007). These agents may have multiple high-level goals from which to choose, but if two or more agents need to coordinate their actions, the agents' meta-control components must be on the same page. That is, the agents must reason about the same problem and may need to be at the same stage of the problem-solving process (e.g., if one agent decides to devote little time to communication/negotiation before moving to other deliberative decisions while another agent sets aside a large portion of deliberation time for negotiation, the latter agent would waste time trying to negotiate with an unwilling partner). Thus if an agent changes the problem solving context it is focusing on, it must notify other agents with which it may interact. This suggests that the meta-control component of each agent should have a multi-agent policy, where the progression of what deliberations agents do, and when, is choreographed carefully and includes branches to account for what could happen as deliberation (and execution) plays out.

Determining the multi-agent policy is a complicated problem since the multi-agent policy is not expected to be simply the union of all single-agent meta-control policies. Consider for instance, two agents A1 and A2 are negotiating about when A1 can complete method M_1 that enables A2's method M_2. This negotiation involves an iterative process of proposals and counter-proposals where at each stage A2 generates a commitment request to A1, A1 performs local optimization computations (scheduling) to evaluate commitment requests; this process repeats until A1 and A2 arrive at a mutually acceptable commitment. The meta-level control decision would be to ensure that A1 completes its local optimization in an acceptable amount of time so that A2 can choose alternate methods in case the commitment is not possible. In setting up a negotiation, the meta-level control should establish when negotiation results will be available. This involves defining important parameters of the negotiation including the earliest time the target method will be enabled. Two agents with different views of meta-control policy for negotiation need to be reconciled in order to set up the earliest starting time parameter used in the negotiation process.

A second multi-agent meta-level control research issue involves exploring how to dynamically split the agent network into neighborhoods that are coordinated but not necessarily

the same. Coordinated meta-level control decisions do not mean that meta-level control has to be the same in all parts of the network; instead, it involves finding consistent sets for different parts of the network. A third issue involves problem-solving contexts, which contain agent state information and other data required for decision making. We identify two types of contexts: 1) current context, which is the agents context in the midst of execution; and 2) pending context, where an agent deliberates about various what-if questions related to coordination with other agents. At any time, an agent has one current context and may have one or more pending contexts. When an agent is assigned a task, it creates a pending context where deliberative activities such as negotiation, local scheduling, and policy computation are performed. The meta-level control component will have to determine which pending context to deliberate about and when to switch from executing current context and deliberate about a pending context.

Multi-agent meta-control suggests the need for some kind of meta-level message passing. There are important tradeoffs between the amount of communication (both size and number of messages) and resulting overhead, and the usefulness of such communication. Agents must determine what kind of information is contained in a meta-level message. In some situations, it may be enough for the agent to simply let others know that it is thinking about context X; in other cases, such as when agents are more tightly coupled, an agent may need to communicate some partial results of its current thinking as well. Agents must also reason about how to handle meta-control messages from others and coordinate when these messages should be received and handled

Relevance to a Real-World Application

The following is a description of a real-world application that will need this type of meta-level control. NetRads (Krainin, An, and Lesser 2007; Zink et al. 2005) is a network of adaptive radars controlled by a collection of Meteorological Command and Control (MCC) agents that instruct where to scan based on emerging weather conditions. The NetRad radar is designed to quickly detect low-lying meteorological phenomena such as tornadoes and each radar belongs to exactly one MCC. The MCC agent gathers raw data from the radars and runs detection algorithms on weather data to recognize significant meteorological phenomenon. The time allotted to the radar and its control systems for data gathering and analysis is known as a heartbeat. Results are used to determine potential weather scanning tasks for the next scanning cycle. Tasks are classified as internal tasks (they are squarely under the purview of one agent only) and boundary tasks (shared by multiple MCCs). It is important for MCCs to coordinate both to avoid wasteful redundant scanning as well as to ensure multiple radar scans when required.

Each MCC communicates with its neighbors to agree on what weather scanning tasks it should do and how these tasks should be done in concert with tasks of the neighbors. The MCC then uses a three-step decentralized process to determine the best set of scans for the available radars that will maximize the sum of the utilities of the chosen tasks. It executes a local combinatorial optimization algorithm to determine the best configuration from a local point of view, then exchanges these configurations with neighborhood agents and a hill-climbing negotiation algorithm to determine which radars to activate and how much time to allocate to each task. This process of local optimization and negotiation is time-bounded since radars need to be constantly repositioned to track weather phenomena and recognize the arrival of ones.

In the NetRads domain, an example domain action would be a radar scan of a weather task. The deliberative action in each heartbeat would be the MCC spending some initial time in processing the radar data obtained during the last heartbeat, then performing a local optimization to determine the configuration of the radars under its control, followed by negotiation rounds of alternating communication and recomputation of the local configuration. The meta-level control component guides the optimization and negotiation actions of the MCC.

The main research questions to be addressed in this work are: How to make meta-level control decisions about deliberations and problem solving contexts? How to coordinate the meta-level decision making process among agents? How to ensure that meta-level control has low-overhead? How to dynamically split the network into neighborhoods with varying heartbeats? Which data to collect for performance profiles? How to handle multi-agent meta-level control messages? How to capture and reason about the sequential nature of these research issues? In the next section, we propose a domain-independent framework that addresses these questions.

Solution Approach

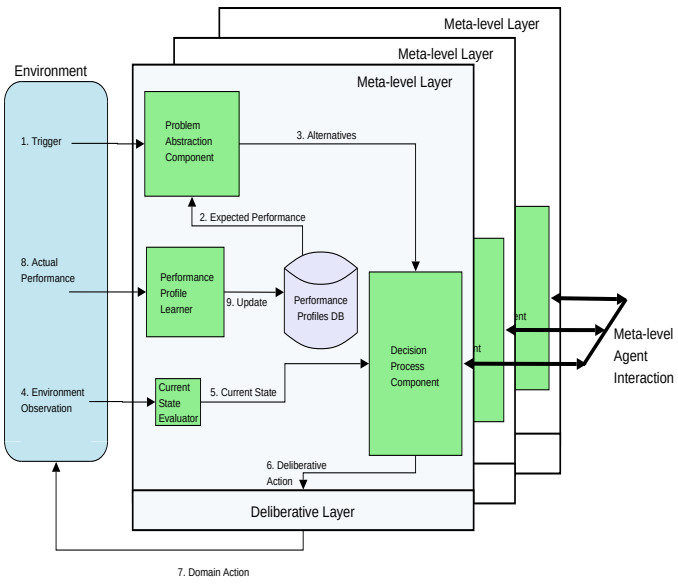

Figure 2: Control Flow in the Generalized Meta-level Control (GeMEC) Architecture)

The Generalized Meta-level Control (GeMEC) architecture (Figure 2) consists of a Problem Abstraction component, a Performance Profile Learner, a Decision Process component and a Current State Evaluator. A trigger is an event requiring the attention of the meta-level layer. It could be the arrival of a new task or a change in the environment. In Netrads, a triggering event occurs at every heartbeat since emerging weather events need to be scanned and analyzed at each heartbeat.

When the meta-level layer is triggered, the Problem Abstraction Component (PAC) extracts the associated task and identifies high-level alternative ways by which the task can be completed successfully. These alternatives are captured in a task structure called Meta-Alt task structure. The PAC uses performance profile (Dean and Boddy 1988; Hansen and Zilberstein 1996) information about the deliberative action modes from a Performance Profile database to determine these alternatives. The PAC will help control the complexity of the meta-level decision process by weeding out all superfluous information. This is especially relevant in applications where each deliberation action can be performed in different modes resulting in different performance characteristics. For instance, scheduling as a deliberation action can have multiple modes: heuristic scheduler (Wagner, Garvey, and Lesser 1997), MDP-based scheduler (Musliner et al. 2006), constraint-based scheduler (Smith et al. 2006) etc.

The GeMEC framework is an extension of Cox and Raja's (Cox and Raja 2007) model that describes meta-level reasoning among multiple agents (Figure 3). Using GeMEC, we elaborate the functionality of the meta-level reasoning module and propose a method for communication as well as distributed decision making among the agents.

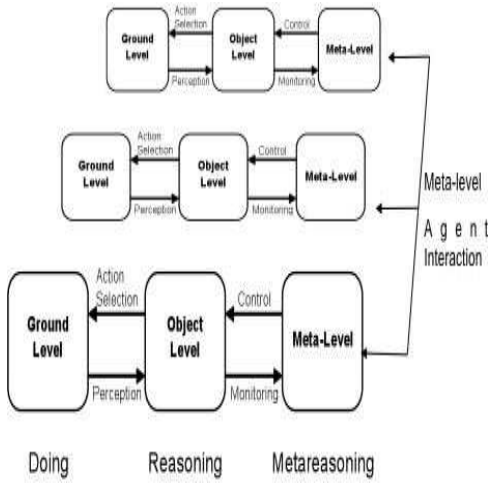

Figure 3: Cox and Raja's Model for multi-agent meta-level reasoning

In the Netrads application, an MCC has multiple deliberation modes. Different amounts of time and resources can be allocated to the heartbeat and within a heartbeat, different allocations to optimization and negotiation cycles can be made. Figure 4 is an example of a task structure representing a set of meta-level alternatives and their associated quality distributions. 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 model quality. The meta-level alternative task structure in Figure 4 describes the agent choices for a particular Netrads tornado tracking scenario. The agent can choose a 30 second or 60 second heart beat. For each choice of heartbeat duration, the agent can also choose the percentage of the heartbeat duration that should be spent on local optimization, assuming that the remaining time is spent on negotiation. These choices for local optimization duration are represented as leaf nodes in the figure and their associated quality distributions are specified. For example, choosing the leaf node *Opt=30%* belonging to the 30 second heartbeat choice indicates that 30% of 30 seconds (9 seconds) will be spent on local optimization and the remaining 21 seconds will be spent on negotiation and this will result in a quality value of 10, 60% of the time and a quality value of 8, 40% of the time.

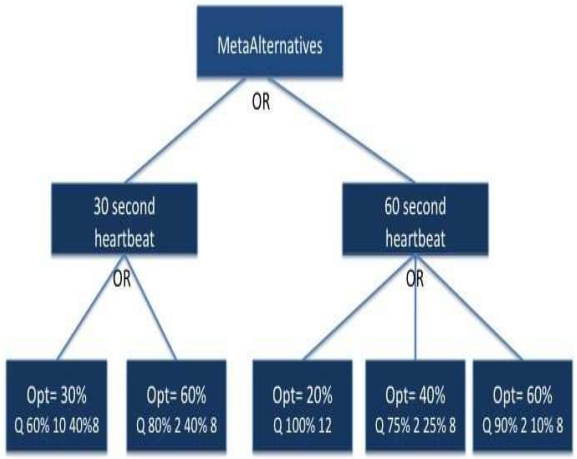

Figure 4: Task structure describing Meta-level alternatives for a specific Netrads scenario

Once constructed, the Meta-Alt task structure is then sent to the Decision Process Component (DPC). In the context of the Netrads application, the DPC converts the meta-level alternative task structure into a Markov Decision Process (MDP) (Bertsekas and Tsitsiklis 1996) using a previously developed algorithm (Raja and Lesser 2007; Alexander and Raja 2006; Alexander, Raja, and Musliner 2008). DPC then computes the policy for the MDP and determines the best deliberative action to recommend to the deliberative layer given the current context.

An MDP-based decision process component is appropriate for the Netrads application since the Meta-Alt task structure is expected to remain static in this application for extended periods of time. Also, the actions of the MCC may not typically be adjusted from heartbeat to heartbeat. In our

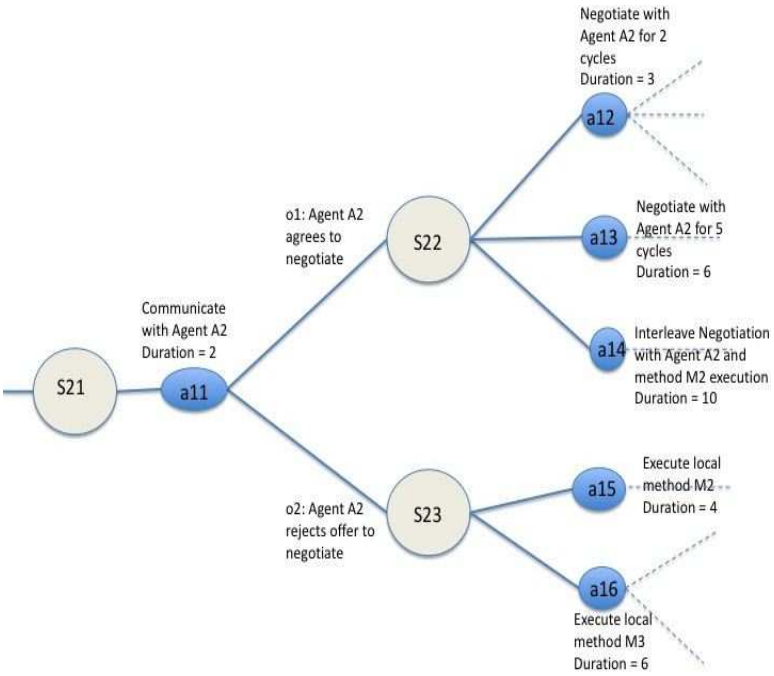

Figure 5: Snapshot of MDP capturing Agent A1's meta-level decision process

view, this justifies embedding the non-trivial cost of computing an optimal or near-optimal policy for the MDP inside the meta-level reasoning loop. In applications with greater level of dynamics, GeMEC's decision process component could be replaced by other decision processing technologies such as stochastic online optimization methods (Bent and Hentenryck 2004). The choice of the decision process component mechanism depends on the dynamics of the application and the cost of computing the action choices.

Procedure 1 summarizes the reasoning process:

Procedure 1 GeMEC Reasoning loop for Netrads

1: extract task
2: use Performance Profile database to generate Meta-Alt task structure
3: send Meta-Alt task structure to DPC
4: convert Meta-Alt task structure to a MDP
5: derive optimal policy and recommend best action choice

In Netrads, the DPC could adjust the system heartbeat to adapt to changing weather conditions. For example, if many scanning tasks are occurring in a certain region, meta-control may decide to use a shorter heartbeat to allow the system to respond more rapidly. It is important to handle different heartbeats for different neighborhoods of MCCs. For example, suppose there is a neighborhood with a heartbeat of 30 seconds and another one with a heartbeat of 60 seconds, with one MCC belonging to both neighborhoods. Also suppose that this MCC is using a 30-second heartbeat. We can choose between one of two protocols: After the end of its first heartbeat, new information is sent to its neighbors in the 60-second neighborhood, or the agent negotiates only with its 30-second neighbors until its slower neighbors have entered their next heartbeat.

The DPC can also adjust the parameters involved in the calculation of an MCC's local configuration in order to trade off optimality for a shorter run time, which would allow more rounds of negotiation to be performed. Spending more time on negotiation may be preferable, for example, when there are many boundary tasks compared to internal tasks. In fact, the DPC could make this decision for each round in the negotiation process. This would allow a fast estimate of the optimal local configuration to begin negotiation and then switch to a better optimization in later rounds, or perhaps begin with the current brute force optimization and switch to other methods after negotiating for a while.

Our approach to the DPC involves formalizing the agent's meta-level control problem using a MDP (Alexander and Raja 2006; Raja and Lesser 2007). The advantage of using a MDP to model the meta-level control problem is two-fold : the MDP supports sequential decision making allowing for non-myopic decision making; and meta-level reasoning will be inexpensive as it will involve only MDP policy lookup. Several researchers (Peshkin et al. 2000; Nair et al. 2003) have studied multi-agent MDPs but most have assumed full (individual) observability. However, like Becker et. al. (Becker et al. 2003), the Netrads application involves agents having partial and different views of the global state. Decentralized Markov Decision Process (DEC-MDP) (Bernstein, Zilberstein, and Immerman 2000) is a suitable framework for these types of problems. A DEC-MDP is a DEC-POMDP that is jointly fully observ-

able (i.e., the combination of both agents observations determine the global state of the system). The complexity of DEC-MDPs has been shown to be NEXP-complete for finite horizon problems and undecidable for infinite-horizon problems (Bernstein, Zilberstein, and Immerman 2000). The complexity is mainly due to the explosion in states, action choices and observations due to agent interactions.

One way to overcome this complexity is to approximate the solution. An approximation of particular interest (Goldman and Zilberstein 2005) leverages the idea that a lot of decentralized problems have some structure with influence on the level of decentralization. The key idea is that a feasible approximation can be found by decomposing the global reward function into local and temporal problems. A communication policy is then used to synchronize the single-agent MDPs occasionally. This would allow agents to exchange information at certain times to obtain global state (Xuan, Lesser, and Zilberstein 2001). When not communicating, the agents will act independently of each other and not necessarily follow the optimal policy. This local policy of an agent can be viewed as Sutton's options (Sutton, Precup, and Singh 1999). The options will have terminal actions, specifically communication actions. In this model, it is also assumed that all options are terminated whenever atleast one of the agents initiates communications. It is also assumed that when the agents exchange information the global state of the system is revealed.

This model is shown to be equivalent to a multi-agent markov decision process (MMDP) (Goldman and Zilberstein 2005). The optimal policy would involve searching all over possible pairs of local single agent policies and communication policies. For example, each agent needs to ensure that it enters into a negotiation mode only upon agreement of the partner agent to enter into the similar mode. If the other agent refuses to negotiate, the requesting agent should make sure that it does not enter in to a negotiation mode. This behavior will be manifested by an appropriate definition of the reward function and will enforce *coordinated meta-level control*. The reward function for each state *s* will have two components and can be defined as: $R(s) = R_{local} + R_{coord}$ where R_{local} is the reward obtained from executing local actions and the R_{coord} is the reward obtained from actions requiring coordination. This reward function has to capture the fact that an agent shouldn't consider doing negotiation as an action choice for a particular time frame unless the other agent is choosing the negotiation action for an overlapping time frame.

In addition to agreeing to negotiate, the agents have to be on the same page as far negotiation parameters are concerned. Each agent also needs to determine how much time/resources to allocate to the negotiation process. The motivation for longer durations would be to improve the success rate of negotiation. The agent can also choose to interleave the negotiation with a local domain action execution so that the negotiation has more end-to-end time, thus accounting for uncertainty in outcome which improves the success probability of the negotiation.

We now describe a simple example of DPC's decision making process. Consider a scenario where there are two agents A1 and A2 with an interdependency requiring negotiation. Figure 5 describes the meta-level MDP for agent A1. State *S21* is a state where the agent has to communicate with agent A2 and determine whether A2 is willing to negotiate in the next time period. This is represented by action *a11* which has a duration of 2 units. Agent A2 can respond with one of two responses: agree to negotiate (outcome o1) or refuse to negotiate (outcome o2). The states reached and action choices available to agent A1 vary depending on the outcome of action *a11*. The actions available from state *S22* allow agent A1 to allocate different amounts of time to negotiation where the entire processor is allocated to negotiation only (durations 3 and 6 for actions *a12* and *a13* respectively) or the agent can choose to interleave the negotiation with a domain action (method *M2*) for a duration of 10 units (action *a14*). If outcome o2 occurs, agent A1 will not move into a negotiation phase and instead try to ensure that it uses its resources efficiently by choosing one of its local domain actions (actions *a15* and *a16*).

Conclusions and Future Work

In this paper we map the various multi-agent meta-level questions to a single generalized formalization of meta-level control; we then elaborate how these issues will be addressed in a real tornado tracking application; finally, we describe a methodology to construct a class of MDPs with the ability to model interactions among multiple meta-level decision process components. Our next step is to further elaborate the details of the example and study the appropriateness for all the meta-level issues proposed in this paper. We will also collect performance profiles for various scenarios in the Netrads application and empirically verify our MDP based-model and study the value for explicit versus implicit communication among agents.

Acknowledgments

We thank George Alexander, Michael Krainin and Jason Hillgen for their assistance in studying the multi-agent meta-level control problem in the context of Netrads. We thank the reviewers for their helpful feedback.

References

Alexander, G., and Raja, A. 2006. The role of problem classification in online meta-cognition. In *Proceedings of 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 218–225.

Alexander, G.; Raja, A.; Durfee, E.; and Musliner, D. 2007. Design paradigms for meta-control in multi-agent systems. In *Proceedings of AAMAS 2007 Workshop on Metareasoning in Agent-based Systems*, 92–103.

Alexander, G.; Raja, A.; and Musliner, D. 2008. Controlling deliberation in a markov decision process-based agent. In *To appear Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems*.

Becker, R.; Zilberstein, S.; Lesser, V.; and Goldman, C. V. 2003. Transition-Independent Decentralized Markov Decision Processes. In *Proceedings of the Second International*

Joint Conference on Autonomous Agents and Multi Agent Systems, 41–48. Melbourne, Australia: ACM Press.

Bent, R., and Hentenryck, P. V. 2004. Regrets Only! Online Stochastic Optimization Under Time Constraints. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-2004)*, 501–506.

Bernstein, D.; Zilberstein, S.; and Immerman, N. 2000. The Complexity of Decentralized Control of Markov Decision Processes. In *In Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, 32–37.

Bertsekas, D. P., and Tsitsiklis, J. N. 1996. *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific.

Cox, M. T., and Raja, A. 2007. Metareasoning: A Manifesto. Technical Report BBN TM-2028, BBN Technologies.

Cox, M. T. 2005. Metacognition in computation: a selected research review. *Artif. Intell.* 169(2):104–141.

Dean, T., and Boddy, M. 1988. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, 49–54. Saint Paul, Minnesota, USA: AAAI Press/MIT Press.

Doyle, J. 1983. What is rational psychology? toward a modern mental philosophy. *AI Magazine* 4(3):50–53.

Goldman, C., and Zilberstein, S. 2005. Goal-oriented dec-mdps with direct communication. Computer Science Technical Report TR-04-44, University of Massachusetts at Amherst.

Hansen, E., and Zilberstein, S. 1996. Monitoring the Progress of Anytime Problem-Solving. In *Proceedings of the 13th National Conference on Artificial Intelligence*, 1229–1234.

Horvitz, E. J. 1988. Reasoning under varying and uncertain resource constraints. In *National Conference on Artificial Intelligence of the American Association for AI (AAAI-88)*, 111–116.

Krainin, M.; An, B.; and Lesser, V. 2007. An Application of Automated Negotiation to Distributed Task Allocation. In *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007)*, 138–145. Fremont, California: IEEE Computer Society Press.

Musliner, D.; Durfee, E.; Wu, J.; Dolgov, D.; Goldman, R.; and Boddy, M. 2006. Coordinated plan management using multiagent mdps. in distributed plan and schedule management. In *Distributed Plan and Schedule Management: Papers from the 2006 AAAI Spring Symposium*.

Nair, R.; Tambe, M.; Yokoo, M.; Pynadath, D.; and Marsella, S. 2003. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intel ligence (IJCAI)*.

Peshkin, L.; Kim, K.-E.; Meuleau, N.; and Kaelbling, L. P. 2000. Learning to cooperate via policy search. In *Sixteenth Conference on Uncertainty in Artificial Intelligence*, 307–314. San Francisco, CA: Morgan Kaufmann.

Raja, A., and Lesser, V. 2007. A framework for meta-level control in multi-agent systems. *Autonomous Agents and Multi-Agent Systems* 15(2):147–196.

Raja, A. 2003. Meta-level control in multi-agent systems. *PhD Thesis, Computer Science Department, University of Massachus etts at Amherst*.

Russell, S., and Wefald, E. 1989. Principles of metareasoning. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, 400–411.

Russell, S. J.; Subramanian, D.; and Parr, R. 1993. Provably bounded optimal agents. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, 338–344.

Schut, M., and Wooldridge, M. 2001. The control of reasoning in resource-bounded agents. *Knowledge Engineering Review* 16(3):215–240.

Simon, H., and Kadane, J. 1974. Optimal problemsolving search: All-or-nothing solutions. Computer Science Technical Report CMU-CS-74-41, Carnegie Mellon University.

Smith, S.; Gallagher, A.; Zimmerman, T.; and Barbulescu, L. 2006. Multi-Agent Management of Joint Schedules. In *In Distributed Plan and Schedule Management: Papers from the 2006 AAAI Spring Symposium*.

Stefik, M. 1981. Planning and meta-planning. *Artificial Intelligence* 16(2):141–170.

Sutton, R., and Barto, A. 1998. *Reinforcement Learning*. MIT Press.

Sutton, R. S.; Precup, D.; and Singh, S. P. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112(1-2):181–211.

Wagner, T.; Garvey, A.; and Lesser, V. 1997. Criteria-Directed Heuristic Task Scheduling. UMASS Department of Computer Science Technical Report TR-97-16.

Xuan, P.; Lesser, V.; and Zilberstein, S. 2001. Communication Decisions in Multi-agent Cooperation: Model and Experiments. *Proceedings of the Fifth International Conference on Autonomous Agents* 616–623.

Zink, M.; Westbrook, D.; Abdallah, S.; Horling, B.; Lyons, E.; Lakamraju, V.; Manfredi, V.; Kurose, J.; and Hondl, K. 2005. Meteorological Command and Control: An End-to-end Architecture for a Hazardous Weather Detection Sensor Network. In *Proceedings of the ACM Workshop on End-to-End, Sense-and-Respond Systems, Applications, and Services (EESR 05)*, 37–42.