Real-Time Meta-Level Control in Multi-Agent Systems *

Anita Raja and Victor Lesser

Department of Computer Science,
University of Massachusetts,
Amherst, MA 01003-4610, USA
{araja, lesser}@cs.umass.edu

Abstract. Open environments are characterized by their uncertainty and non-determinism. Sophisticated agents operating in these environments must reason about their local problem solving activities, interact with other agents, plan a course of action and carry out actions in the face of limited resources and uncertainty about action outcomes and the actions of other agents, all in real-time. Many efficient architectures and algorithms that support these activities have been developed and studied. However, none of these architectures explicitly reason about the consumption of time and other resources by control activities such as scheduling and coordination, which may degrade an agent's performance. This paper describes work in progress to define and build a meta-level control framework which will account for the cost of making appropriate decisions about these control activities, without consuming significant resources in the process.

1 Introduction

Agents perform local planning and scheduling and coordinate with other agents to improve their performance. Many efficient architectures and algorithms[3, 1, 5] that support these activities have been developed and studied. However, none of these architectures explicitly reason about the time and other resources consumed by these control activities, which may in fact degrade an agent's performance. An agent is not performing rationally if by the time it has calculated an action, it is no longer applicable. Hence, an agent should only plan and/or coordinate when the expected improvement outweighs the expected cost. If significant resources are expended on making this meta-decision, then the meta-meta decisions on whether to spend these resources should be made in the context of the overall gain to the system utility. To do this an agent would have to know the effect of all combinations of actions ahead of time, which is intractable for any

* This material is based upon work supported by the the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory Air Force Materiel Command, USAF, under agreement number F30602-99-2-0525 and the National Science Foundation under Grant No. IIS-9812755 and IRI-9634938.

reasonably sized problem. The problem of how to approximate this ideal of sequencing domain and control activities without consuming too many resources in the process, is the **meta-level control problem** for a resource bounded rational agent.

Agent activities can be broadly classified into three categories - domain, control, and meta-level control activities. Domain activities are executable primitive actions which achieve the various high-level tasks. Control activities are of two types, scheduling activities which choose the high level goals, set constraints on how to achieve them and sequence the domain level activities; and coordination activities which facilitate cooperation with other agents in order to achieve the high-level goals. The meta-level control activities optimize the agent's performance by allocating appropriate amount of processor and other resources at appropriate times to the domain and control activities.

There are two main questions addressed in this work. The first is how to account for the overhead of control activities? Specifically, how should the system account for rescheduling and coordination activities when determining action sequences and related commitment times? The second is how to integrate meta-level activities with domain and control activities? In other words, how to make choices about how much effort to spend on meta-level control, control and domain activities respectively so that overall system performance is optimized?

The following domain assumptions are made in the work described here : The agents are cooperative and will prefer alternatives which increase social utility even if it is at the cost of decreasing local utility. However, the solution approach proposed here generalizes to self-interested environments too. An agent may concurrently pursue multiple high-level goals and the completion of a goal derives utility for the system or agent. The overall goal of the system or agent is to maximize the utility generated. Quality and Utility are equivalent measures of performance criteria in this system. The high-level goals are generated by either internal or external events being sensed and/or requests by other agents for assistance. It is not necessary for all high-level goals to be completed in order for an agent to derive utility from its activities. The partial satisfaction of high-level goal is sometimes permissible while trading-off utility derived for resource usage. Scheduling and coordination activities do not have to be done immediately after there are requests for them and in some cases may not be done at all. There are alternative ways of completing scheduling and coordination activities which trade-off the likelihood of these activities resulting in optimal decisions versus the amount of resources used.

The remainder of this document is structured as follows: The next section describes the taxonomy of decisions for a single agent in a multi-agent scenario. In Section 3, we briefly discuss how we anticipate solving the meta-level control problem. In Section 4 we report on the progress and expected implementation developments of this work.

2 Taxonomy of Decisions

In this section, we provide a high level classification of the meta-level issues which motivate a decision-theoretic framework for meta-level control. There are two types of decisions made by an agent: the meta or macro-level decisions handled by the meta-level controller and the scheduling or micro-level decisions handled by the domain-level controller. Figure 1 describes the hierarchy of decisions. The meta-level controller will be designed to make quick and inexpensive decisions on how much resources should be spent on domain versus control actions. The initial control decisions are further classified into **Coordination decisions** which dictate whether or not to coordinate with other agents and how much effort should be spent on coordination, **Scheduler decisions** which dictate whether or not to call the domain-level scheduler and how much effort should be spent by the scheduler and **Slack decisions** which will prescribe how much total slack/free time should be included in a schedule to deal with unexpected events. Coordination in this work is the inter-agent negotiation process which establishes commitments on completion times of tasks or methods

An example of a coordination meta-level decision is determining how long a negotiation should take. If an agent decides to negotiate, it should also decide whether to negotiate by means of a single step or a multi-step protocol that may require a number of negotiation cycles to find an acceptable solution or even a more expensive search for a near-optimal solution. The benefit of choosing a more expensive protocol is that on average, an agent receives high utility as a result of successfully completing the negotiation as expected. The cost involved is that more resources are invested in negotiation, specifically more end-to-end time in case of multi-step negotiation and higher computation power and time in case of near-optimal solutions.

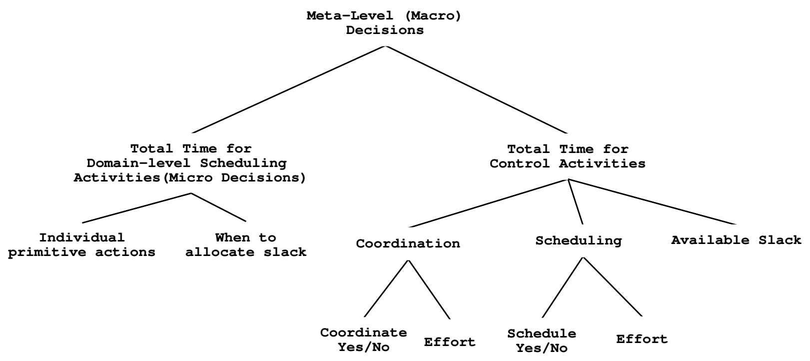

Fig. 1. Decision space for a single agent

The high-level goal of this work is to create agents which can perform useful tasks. These agents necessarily have limited computation and the model of the task environments are not readily available. Reinforcement learning is useful for learning the utility of these control activities and decision strategies in different contexts. Hence we are naturally led to build a meta-level controller which uses reinforcement learning techniques to allocate computation effectively. A basic assumption of our approach is to build a meta-level controller for a specific environment rather than handle any arbitrary environment. The search space for each environment is represented using factored Markov Decision Processes(MDPs). Factored MDPs are compact representations of MDPs using Bayesian Networks.

3 Solution Methodology

In this section, we outline our approach for providing effective meta-level control for resource-bounded agents. We begin by describing the architecture of a resource-bounded agent that uses offline policies built using reinforcement learning. We discuss how the overhead associated with meta-level control and control activities is explicitly costed out. We also enumerate the characteristics of the policy which allow for the agent to dynamically react to real-time unanticipated exogenous events.

A conceptual architecture of a resource-bounded agent that uses offline-learning for meta-level control is illustrated in figure 2. A possible flow of control is described by the number sequences in bold. At the heart of the system is the domain problem solver(DPS). It receives tasks and other exogenous requests from the environment. It then sends the task set, resource constraints and criteria to the meta-level controller. The controller sends the best policy which has been computed offline for the given constraints back to the DPS. The DPS sends the policy to the execution component where execution begins. The DPS interfaces with the scheduler and coordination components and invokes them as needed.

We account for the cost of all three levels of the decision hierarchy - meta-level control, control and domain activities. Meta-level control involves reasoning about the cost of negotiation and scheduling activities. The cost of this reasoning is accounted for directly within the MDP framework. Negotiation costs are reasoned about explicitly in our framework since they are modeled in the task structures, which describe alternate ways of achieving the task. The negotiation tasks are split into an information gathering phase and a negotiating phase, with the outcome of the former enabling the latter. The negotiation phase can be achieved by choosing one of two protocols, which differ in their performance characteristics. The information gathering phase and negotiation methods are modeled as individual primitive actions in the task structure. Thus reasoning about the costs of negotiation is done explicitly just as it is done for regular domain-level activities.

However, the costs associated with reasoning and scheduling of unanticipated exogenous events are not dealt with explicitly. Instead, the costs are accounted by

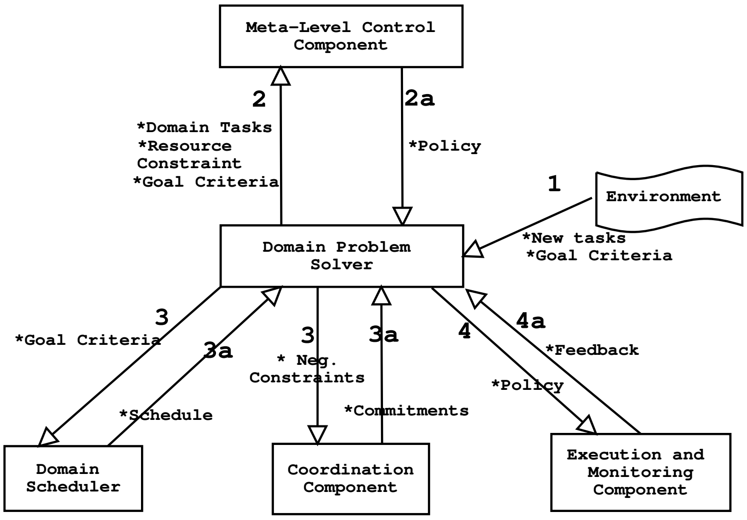

Fig. 2. Conceptual architecture of a bounded rational agent

the opportunity cost of using slack in schedules as well as the cost of computing complex state features for the Markov Decision Process which models the search space of the meta-level controller.

Slack in the execution policy provides flexibility for the policy to dynamically react to unexpected events. The flexibility is built-in at the cost of diminished expected performance characteristics. To handle unexpected exogenous events, the system learns to estimate when to allocate slack and how much slack to allocate since slack is one of the input parameters to the domain-level scheduler.

Reasoning about scheduling activities are also costed out by accounting for the overhead of state feature computation. There are two types of state features for the states represented in the search space MDP- **simple** features where the reference values are readily available by simple lookups and **complex** features which involve significant amount of computation to determine their values, which includes invoking the domain-level scheduler. We will make explicit decisions whether to gather complex features and determine which complex features are appropriate. The complex features are usually context-sensitive and involve computations which take time that is sufficiently long that, if not accounted for, will lead to incorrect meta-level decisions. The decision on whether to compute the complex features constitute the meta-meta-level decisions mentioned in the introduction. An instance of a complex feature is that instead of having a feature which gives a general description of the slack distribution in the current schedule i.e. there is a lot of slack in the beginning or end of the schedule, there is a context-sensitive feature which examines the exact characteristics of the new task and makes a determination whether the available slack distribution will likely allow for the new task to be included in the schedule.

There are five situations where meta-level reasoning is desirable - decision on whether to initiate negotiation, decision of what to do with a new task when it arrives, decision on whether to renegotiate in the event of a renege, decision on amount of scheduler effort and decision to spontaneously reschedule based on execution characteristics. Each event is described as an external action in a decision tree. The external action triggers a state change. The response actions (executable or parameter determining) are also modeled in the decision tree. We model this sequential decision making process using a structured MDP which is a intensional representation of the underlying search space. The search space is a linear combination of the decision tree structures representing the search space for each exogenous event.

The features of MDP state required to make effective meta-level control decisions are enumerated in the table 3. There are thirteen features - the first nine are simple features and the last four are complex features.

FeatureID	Feature
F0	Current status of schedule stack and agenda
F1	Relation of quality gain of a task to currently scheduled task set
F2	Relation of deadline of a task to currently scheduled task set
F3	Relation of priority of agenda items to currently scheduled task set
F4	Percent of slack in local schedule
F5	Percent of slack in other agent's schedule
F6	Relation of priority of non-local task to non-locally scheduled task set
F7	Expected Quality of current task at time t
F8	Actual Quality of current task at time t
F9	Expected Rescheduling Cost with respect to a task set
F10	Expected DeCommitment Cost with respect to a particular task
F11	Relation of abstractions of new task to given task
F12	Relation of slack fragments in local schedule to new task
F13	Relation of slack fragments in non-local agent to non-local task

Fig. 3. Table of labels of state features and their description

To provide an intuition of the decision process for an exogenous event, we describe the choices available to the agent and the approximate criteria for making the choice for when the agent has to decide whether to initiate negotiate. For the sake of clarity, every leaf in the decision tree is marked by an identifier and referenced by the associated text description. The following are not rules learned by the system. Instead, we have hand-crafted some intuitive decision rules that could potentially be approximations to the actual rules learned by the system.

Suppose there is a subtask or method Tx in currently scheduled task which either requires a non-local method to be executed as a precondition or should be sub-contracted out to another agent. The local agent has to decide whether it is worth its while to even initiate negotiation and if so, what kind of negotiation mechanism to use. There is a information gathering action called *MetaNeg*, (see Figure 4) which will gather information on the state of the non-local agent that is being considered for negotiation.

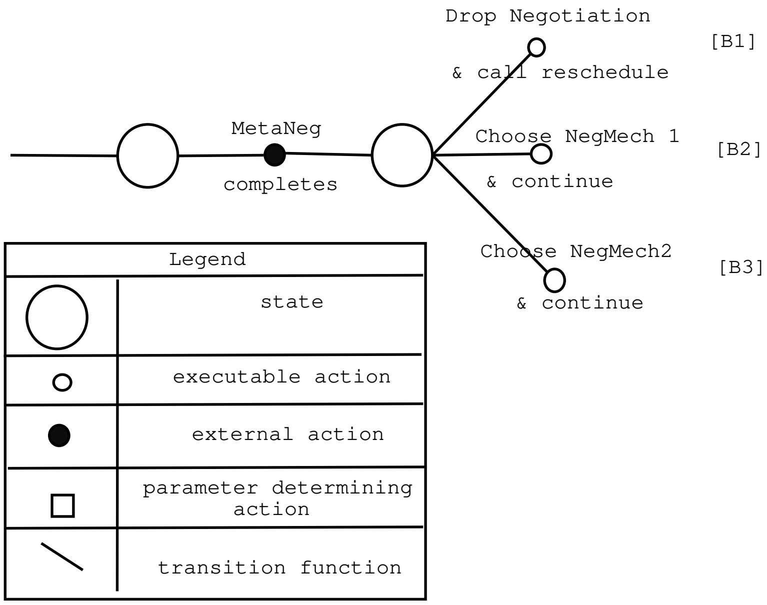

Fig. 4. Decision tree on initiating negotiation and amount of effort

1. No negotiation and call reschedule if needed[**B1**]: If utility of the task set of the non-local agent is very high, task utility of task Tx is relatively low, and there is not enough slack to fit Tx in the non-local schedule, then don't negotiate. The features used are F1, F4, F5 and F6.
2. Negotiate with NegMech1 [**B2**] : If utility of the task set of the other agent is low, flexibility of the schedule of the non-local agent is high and utility for task Tx is high, then the agent should negotiate. The choice of the exact negotiation mechanism will depend on the relative gain from doing task Tx and the actual amount of slack available in the other agent. The complexity of the negotiation mechanism is directly proportional to the value of the negotiated task and availability of slack. NegMech1 is a single-shot negotiation mechanism which works in a all or nothing mode. Its inexpensive but has a lowered probability of success. The features used are F1, F4, F5 and F6.
3. Negotiate with NegMech2 [**B3**] : If utility of the task set of the other agent is low, flexibility of the schedule of the non-local agent is high and utility for task Tx is high, then the agent should negotiate. The criteria for the choice of the exact negotiation mechanism is the same as above. NegMech2 is a

multi-try negotiation mechanism which tries to achieve a commitment by several proposals and counter-proposals until a consensus is reached or time runs out. Its expensive but has a higher probability of success. The features used are F1, F4, F5 and F6.

The search space is constructed for a specific task environment which is characterized by a set of agents with varying task assignments and an arrival model of tasks. The transition probabilities are direct functions of the performance characteristics and sequencing constraints of the primitive actions[5]. The reward function is similar to the brownie point model described in [2]. This is a more appropriate reward function than the intuitive sum of the utilities of the agents in the cooperative MAS, because the latter is expensive to compute. Further details of the approach and implementation are described in [4]. Offline reinforcement learning is used to obtain the optimal policy for this MDP.

4 Future Work

The next stage in this work is to implement the architecture described in Figure 2. Some of the components such as the domain level scheduler and execution and monitoring component are already available. We will first verify whether a reinforcement learning approach is appropriate for this problem. This will include verifying and identifying the appropriate state features which support accurate meta-level decision making. We will compare the performance of the learning-based meta-level control methodology to base-line methods which have no meta-level control, a case-base of intuitive meta-level heuristics and a quasi-optimal policy which is built assuming there is complete knowledge of performance characteristics of the actions and exogenous events ahead of time for a single specific scenario.

References

1. Craig Boutlier. Sequential Optimality and Coordination in Multiagent Systems. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.
2. Alyssa Glass and Barbara Grosz. Socially conscious decision-making. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 217–224, Barcelona, Catalonia, Spain, 2000. ACM Press.
3. David J. Musliner. Plan Execution in Mission-Critical Domains. In *Working Notes of the AAAI Fall Symposium on Plan Execution - Problems and Issues*, 1996.
4. Anita Raja. Towards Bounded- Rationality in Multi-Agent Systems: A Reinforcement-Learning Based Approach. Technical report, May 2001. http://dis.cs.umass.edu/ãraja/proposal.ps.
5. Anita Raja, Victor Lesser, and Thomas Wagner. Toward Robust Agent Control in Open Environments. In *Autonomous Agents 2000, Barcelona, Spain*, July, 2000.