From: AAAI Technical Report WS-02-15. Compilation copyright © 2002, AAAI (www.aaai.org). All rights reserved.

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
Ph: 413-545-3444 Fax: 413-545-1249

Abstract

Sophisticated agents operating in open environments must make complex real-time control decisions on scheduling and coordination of domain actions. These decisions are made in the context of limited resources and uncertainty about outcomes of actions. The question of how to sequence domain and control actions without consuming too many resources in the process, is the meta-level control problem for a resource-bounded rational agent. Our approach is to design and build a meta-level control framework with bounded computational overhead. This framework will support decisions on when to accept, delay or reject a new task, when it is appropriate to negotiate with another agent, whether to renegotiate when a negotiation task fails and how much effort to put into scheduling when reasoning about a new task.

Introduction

Agents in complex environments must reason about their local problem solving actions, interact with other agents, plan a course of action and carry it out. All these have to be done in the face of limited resources and uncertainty about action outcomes in real-time. Furthermore, new tasks can be generated by existing or new agents at any time, thus an agent's deliberation must be interleaved with execution. The planning, scheduling and coordination of tasks are non-trivial, requiring either exponential work, or in practice, a sophisticated scheme that controls the complexity. In this paper, we describe a framework which will provide effective allocation of computation resulting in improved performance of individual agents in a cooperative multi-agent system.

In this framework, agent actions are broadly classified into three categories - **domain**, **control**, and **meta-level control** actions. Domain actions are executable primitive actions which achieve the various high-level tasks. Control actions are of two types, scheduling actions which choose the high level tasks, set constraints on how to achieve them and sequence the detailed domain level actions which achieve the selected tasks; and coordination actions which facilitate cooperation with other agents in order to achieve the high-level tasks. Meta-level control actions optimize the agent's performance by choosing and sequencing domain and control actions.

Copyright © 2002, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

Agents perform control actions to improve their performance. Many efficient architectures and algorithms that support these actions have been developed and studied(1; 3; 4). Agents receive sensations from the environment and respond by performing actions that affect the environment using the effectors. The agent chooses its domain level actions and this might involve invoking the scheduling and coordination modules. Classic agent architectures either overlook the cost of control actions or they assume a fixed and negligible cost and do not explicitly reason about the time and other resources consumed by control actions, which may in fact degrade an agent's performance. An agent is not performing rationally if it fails to account for the overhead of computing a solution. This leads to actions that are without operational significance (5).

Consider an administrative agent which is capable of multiple tasks such as answering the telephone, paying bills and writing reports. It usually takes the agent a significant amount of time to sort out the bills. Suppose the agent does not perform any meta-level reasoning about the importance or urgency of the tasks. It will then spend the same amount of time deciding whether to pick up a ringing phone as it does on deciding which bills to pay. If the agent is equipped with meta-level reasoning capabilities, it will recognize the need to make quicker decisions on whether to answer the phone than on sorting bills since there is external constraint on the ringing phone, namely that the caller could hang up. The agent will make better decisions on answering calls as well as completing its other tasks by dynamically adjusting its decision based on its current state and the incoming task.

Our proposed architecture will support this dynamic adjustment process by introducing resource-bounded meta-level reasoning in agent control. In addition to the meta-level control component, there are various options for invoking the scheduling and coordination components. These options differ in their resource usage and performance. The meta-level control component will decide if, when and how much control activity is necessary for each event sensed by the agent.

Meta-level control actions include allocating appropriate amount of processor and other resources at appropriate times. To do this an agent would have to know the effect of all combinations of actions ahead of time, which is intractable for any reasonably sized problem. The question of how to approximate this ideal of sequencing domain and control actions without consuming too many resources in the process, is the **meta-level control problem** for a resource bounded rational agent.

Our solution to this problem is to construct a MDP-based meta-level controller which uses reinforcement learning(RL) to learn the utility of control actions and decision strategies in resource-bounded contexts. In order to construct such a controller, it is necessary to identify the features which affect the decisions. The paper is structured as follows: we first enumerate the assumptions made in our approach and describe the agent architecture which will provide meta-level control. We then present and evaluate a case-base of hand-generated heuristics for meta-level control. These heuristics will determine the features necessary for the RL meta-level controller to make accurate decisions. Preliminary experimental results illustrating the strength of meta-level control in agent reasoning and the effectiveness of the heuristics are provided.

Assumptions

The following assumptions are made in the framework described in this paper: The agents are cooperative and will prefer alternatives which increase social utility even if it is at the cost of decreasing local utility. Since the environment is cooperative, we can safely assume that the cumulative of meta-level control decisions at the individual agent level represent the meta-level control reasoning process for a multi-agent system. 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 over some finite time horizon. The high-level goals are generated by either internal or external events being sensed and/or requests by other agents for assistance. These goals must often be completed by a certain time in order to achieve any utility. It is not necessary for all high-level goals to be completed in order for an agent to derive utility from its actions. The partial satisfaction of a high-level goal is sometimes permissible while trading-off the amount of utility derived for decrease in resource usage. The agent's scheduling decisions involve choosing which of these high-level goals to pursue and how to go about achieving them. There can be non-local and local dependencies between tasks and methods. Local dependencies are inter-agent while non-local dependencies are intra-agent. These dependencies can be hard or soft precedence relationships. Coordination decisions involve choosing the tasks which require coordination and also which agent to coordinate with and how much effort must be spent on coordination. Scheduling and coordination actions 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. We also make the simplifying assumption that negotiation results are binding and we assume that the agents will not decommit from their contract at later stages.

Agent Architecture

In this section, we provide an overview of our architecture which provides effective meta-level control for bounded rational agents. Figure 1 describes the control flow within this proposed architecture. The number sequences describe the steps in a single flow of control. At the heart of the system is the **Domain Problem Solver**(DPS). It receives tasks and other external requests from the environment(Step **1**). When an exogenous event such as *arrival of a new task* occurs, the DPS sends the corresponding task set, resource constraints as well constraints of other tasks which are being executed, and performance criteria to the meta-level controller(Step **2**). The controller computes the corresponding state and determines the best action prescribed by the hand-generated heuristic policy for that particular task environment. The best action can be one of the following: to call one of the two domain schedulers on a subset of tasks; to gather more information to support the decision process; to drop the new task or to do nothing. The meta-level controller then sends the prescribed best action back to the DPS(Step **2a**).

The DPS, based on the exact nature of the prescribed action, can invoke the **complex scheduler**, **simple scheduler** or **coordination component**(Step **3**) and receives the appropriate output(Step **3a**). If the action is to invoke the complex scheduler, the scheduler component receives the task structure and objective criteria as input and outputs the best satisficing schedule as a sequence of primitive actions. The complex scheduler can also be called to determine the constraints on which a coordination commitment is established. If the meta-level or the domain scheduler prescribe an action that requires establishing a commitment with a non-local agent, then the coordination component is invoked. The coordination component receives a vector of commitments that have to be established and outputs the status of the commitments after coordination completes. The simple scheduler is invoked by the DPS and receives the task structure and goal criteria. It uses pre-computed abstract information of the task to select the appropriate schedule which fits the criteria.

The DPS can invoke the execution component either to execute a single action prescribed by the meta-level controller or a schedule prescribed by the domain-level scheduler(Step **4**). The execution results are sent back to the DPS(Step **4a**) where they are evaluated and if the execution performance deviates from expected performance, the necessary measures are taken by the DPS.

This work accounts for the cost at all three levels of the decision hierarchy - domain, control and meta-level control activities. The cost of domain activities is modeled directly in the task structures which describe the tasks. They are reasoned about by control activities like negotiation and scheduling.

The cost of control activities are reasoned about by the meta-level control activities. Negotiation costs are reasoned about explicitly in this framework since they can be modeled as part of the domain activities needed to complete a high-level goal. The negotiation tasks are split into an information gathering phase and a negotiating phase, with the

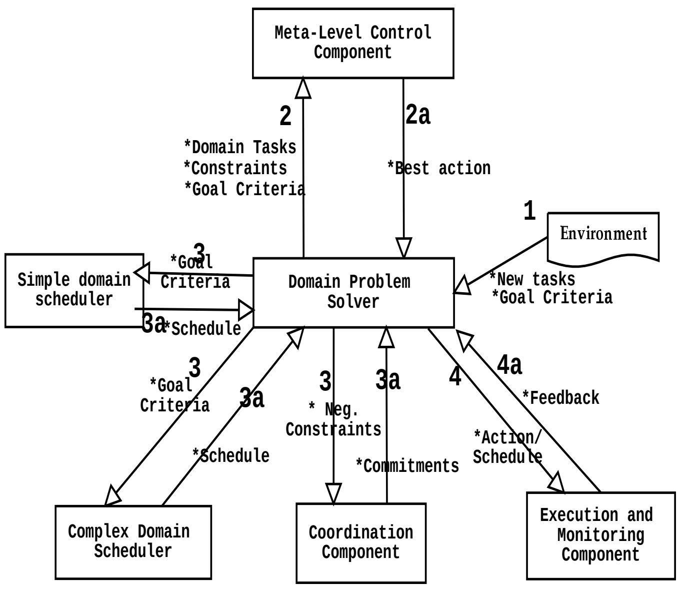

Figure 1: Control-flow in a bounded rational agent

outcome of the former enabling the latter. The negotiation phase can be achieved by choosing a member from a family of negotiation protocols(7). The information gathering phase is modeled as a **MetaNeg** method in the task structure and the negotiation methods are modeled as individual primitive actions. Thus, reasoning about the costs of negotiation is done explicitly, just as it is done for regular domain-level activities. The **MetaNeg** method belongs to a special class of domain actions which request an external agent for a certain set of information and it does not use local processor time. It queries the other agent and returns information on the agent's expected utility from its tasks, expected completion time of its tasks and flexibility of its schedule. This information is used by the meta-level controller to determine the relevant control actions.

However, reasoning about the cost associated with scheduling activities is implicit. A fixed cost is associated with each of the two schedulers and these costs affect the subsequent choice of domain activities made by the control activities. The earliest start time of domain activities are determined by the latest finish times of their corresponding control activities.

Meta-level control activities in this framework are modeled as inexpensive activities. The cost for meta-level control in this framework are incurred by the computation of state features which facilitate the heuristic decision-making process. The state features and their functionality are described in greater detail later on in this section.

The domain level scheduler depicted in the architecture will be an extended version of the Design-to-Criteria(DTC) scheduler(6). Design-to-Criteria (DTC) scheduling is the soft real-time process of finding an execution path through a hierarchical task network such that the resultant schedule meets certain design criteria, such as real-time deadlines, cost limits, and utility preferences. It is the heart of agent control in agent-based systems such as the resource-Bounded Information Gathering agent BIG (2). Casting the language into an action-selecting-sequencing problem, the process is to select a subset of primitive actions from a set of candidate actions, and sequence them, so that the end result is an end-to-end schedule of an agent's activities that meets situation specific design criteria.

We also introduce a simple scheduler based on the use of abstractions of agent task structures. This will support reactive control for highly constrained situations. Abstraction is an offline process where potential schedules and their associated performance characteristics for achieving the high level tasks are discovered for varying objective criteria. This is achieved by systematically searching over the space of objective criteria. Also multiple schedules could potentially be represented by the same abstraction. The abstraction hides the details of these potential schedules and provides only the high level information necessary to make meta-level choices. When an agent has to schedule a task but doesn't have the resources or time to call the complex domain-level scheduler, the generic abstraction information of the task structure can be used to provide the approximate schedule.

Taxonomy of meta-level control decisions

We now describe a taxonomy of the meta-level decisions in a multi-agent system using a simple example scenario. Consider a multi-agent system consisting of 2 agents *A* and *B*. The discussion will focus only on the various meta-level questions that will have to be addressed by agent *A*. *T0* and *T1* are the top-level tasks performed by agent *A*. Each top-level task is decomposed into two executable primitive actions. In order to achieve the task, agent *A* can execute one or both of its primitive actions within the task deadline and the utility accrued for the task will be cumulative (denoted by the *sum* function). Methods are primitive actions which can be scheduled and executed and are characterized by their expected utility, cost and duration distributions. For instance, the utility distribution of method *M2* described as 90% 10 10% 12, indicates that it achieves utility value of 10 with probability 0.9 and utility of 12 with probability 0.1. Utility is a deliberately abstract domain-dependent concept that describes the contribution of a particular action to overall problem solving. There exists an *enables* relationship from task *NX* belonging to agent *B* to method *M2* belonging to agent *A*'s task *T1*. This implies that successful execution of *NX* by agent *B* is a precondition for agent *A* to execute method *M2*.

In the remainder of this section, we enumerate the features computed when the meta-level control component is invoked. The cost of computing and reasoning about these state features reflect the cost of meta-level control reasoning. We then enumerate the various meta-level control decisions and the case-base of heuristics used to make the decisions.

The following are some simple state features which are used in the heuristic decision making process of the meta-level controller. We use qualitative values such as high, medium and low, to represent the various factors which affect the heuristic features. The quantitative values such as quality of 80 versus quality of 60 were classified into qualitative buckets (high versus medium quality) initially based on intuitions on the expected and preferred behavior of the system. They are verified by multiple simulation runs of the system on various test cases.

F0: Current status of system This feature is represented as a 3-tuple representing the NewItems Stack, Agenda and ScheduleStack where each entry in the tuple contains the number of items on the corresponding stack. The new items are the tasks which have just arrived at the agent from the environment. The agenda stack is the set of tasks which have arrived at the agent but whose reasoning has been delayed and they have not been scheduled yet. The schedule stack is the set of tasks currently being scheduled. Eg. $< 2, 0, 1 >$ means there are two new items which have arrived from the environment and there is one task being scheduled.

F1: Relation of utility gain per unit time of a particular task to that of currently scheduled task set: The $utility\ gain\ per\ unit\ time$ of a task is the ratio of $total\ expected\ utility$ to $total\ expected\ duration$ of that task. This feature compares the utility of a particular task to that of the existing task set and helps determine whether the new task is very valuable, moderately valuable or not valuable in terms of utility to the local agent.

F2: Relation of deadline of a particular task to that of currently scheduled task set: This feature compares the deadline of a particular task to that of the existing task set and helps determine whether the new task's deadline is very close, moderately close or far in the future.

F3: Relation of priority of items on agenda to that of currently scheduled task set: This feature compares the average priority of the existing task set to the priority of the new task and helps determine whether the new task is very valuable, moderately valuable or not valuable in terms of utility to the local agent. Priority is a function of the utility and deadlines of the tasks. Computing the average priority of a task set is a more complicated function than computing the priority of a single tasks since it involves recognizing dominance of individual tasks.

The experiments described in this paper use the above four features. There are other features, simple and complex, which are enumerated below but yet to be implemented..

F4: Percent of slack in local schedule: This feature is used to make a quick evaluation of the flexibility in the local schedule. The amount of slack in the local schedule allows the agent to accept new tasks and schedule them in the free slots as well as deal with unexpected meta-level control activities.

F5: Percent of slack in other agent's schedule: This feature is used to make a quick evaluation of the flexibility on the other agent's schedule. The computation of feature F5 is inexpensive since it is done after an information gathering phase, represented by a primitive action called *MetaNeg* which when executed will gather information on non-local agents which are potential coordination partners for the local agent.

F6: Relation of utility gain per unit time of non-local task to non-local agent's current task set: This feature compares the utility of a particular task to that of the existing task set of a non-local agent and helps determine whether the new task is very valuable, moderately valuable or not valuable with respect to the utility of the other agent. The computation of feature F6 is inexpensive since it too is done after the information gathering phase.

F7: Expected utility of current schedule item at current time: This is the expected utility of the current schedule item at time *t* as determined by the domain-level scheduler which uses expected value computations.

F8: Actual utility of current schedule item at current time: This is the actual quality of the current schedule item at run time t. This feature is compared to F7 in order to determine whether schedule execution is proceeding as expected. If it is not proceeding as expected, a reschedule is initiated to prevent the agent from reaching a failure point from which recovery is not possible. Features F7 and F8 will be computed at specified monitoring points.

F9: Expected Rescheduling Cost with respect to a task set: This feature estimates the cost of rescheduling a task set and it depends on the size and quality accumulation factor of the task structure. It also depends on the horizon and effort parameters specified to the domain-level scheduler.

F10: Expected DeCommitment Cost with respect to a particular task: This is a complex feature which estimates the cost of decommiting from a method/task by considering the local and non-local down-stream effects of such a decommit. The domain-level scheduler could be invoked a number of times to compute this feature making it expensive to compute.

F11: Relation of slack fragmentation in local schedule to new task: This is a complex feature which determines the feasibility of fitting a new task given the detailed fragmentation of slack in a particular schedule. It involves resolving detailed timing and placement issues.

F12: Relation of slack fragments in non-local agent to non-local task: This is a complex feature which determines the feasibility of fitting a new task given the detailed fragmentation of slack in a particular non-local schedule. It involves resolving detailed timing and placement issues.

The following are some of the specific meta-level decisions that will be addressed by any individual agent. We describe how the heuristics determine the best action when certain exogenous events occur. The description is limited to reasoning about features F0-F4. Current work allows for reasoning about all 12 features.

1. Arrival of a new task from the environment: When a new task arrives at the agent, the meta-level control component has to decide whether to reason about it later; drop the task completely; or to do scheduling-related reasoning about an incoming task at arrival time and if so, what type of scheduling - complex or simple. The decision tree describing the various action choices named A1-A9 is shown in Figure 2. Each of the meta-level decisions have an associated decision tree. As each exogenous event occurs for a particular environment, its corresponding decision tree is added incrementally to the parent MDP for that environment and the optimal policy will be computed offline. **Heuristic Rule:** If the new task has very low or

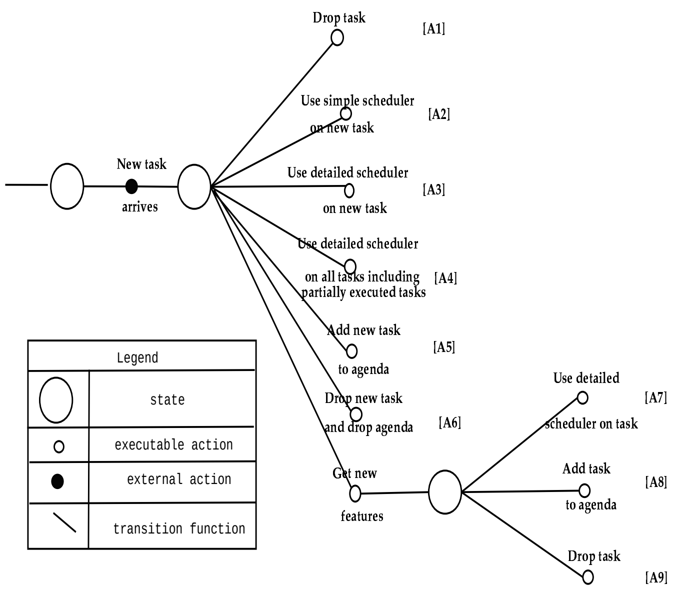

Figure 2: Decision tree when a new task arrives

negligible priority and high opportunity cost with respect to taking resources away from future higher priority tasks, then it should be discarded. If the incoming task has very high priority, in other words, the expected utility is very high and it has a relatively close deadline, then the agent should override its current schedule and schedule the new task immediately. If the deadline is very tight the agent will uses the abstraction-based simple scheduler; else, it will use the more complex scheduler. If the current schedule has average utility that is significantly higher than the new task and the average deadline of the current schedule is significantly closer than that of the new task, then reasoning about the new task should be postponed till later. If the new task is scheduled immediately, the scheduling action costs time, and there are associated costs of dropping established commitments if the previous schedule is significantly revised or completely dropped. These costs are diminished or avoided completely if the task reasoning is postponed to later or completely avoided if the task is dropped.

2. Decision on whether to negotiate: The meta level controller will decide to negotiate based on the information returned by the **MetaNeg** action. It queries the other agent and returns information on the agent's expected utility from its tasks, expected completion time of its tasks and flexibility of its schedule. We know that method *M2* in agent *A* is enabled by task *NX* belonging to agent *B*. The benefit from including method *M2* in agent *A's* schedule is that it increases its total utility. However, it also requires agent *A* and *B* to negotiate over the completion time of task *NX* by agent *B* and this negotiation has an associated cost as well as there is a resource cost to the agent which agrees to the contract. **Heuristic Rule:** If the other agent's current expected utility is much lower than the results of the negotiation, then the local agent will initiate negotiation. Negotiation is also initiated if the other agent's tasks have high utility but the deadlines are far enough in the future to permit the other agent to execute the enabling task. If the other agent's tasks have higher priority than the local task, then the negotiation option is dropped.

3. Choice of negotiation protocol: When 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 single shot protocol is quick but has a higher chance of failure where as a more complex protocol takes more time and has a higher chance of success **Heuristic Rule:** If the agent receives high utility from the results of the negotiation, then the agent should choose the more effective albeit more expensive protocol. The protocol which has a higher guarantee of success require more resources, more cycles and more end-to-end time in case of multi-step negotiation and higher computation power and time in case of near-optimal solutions. (The end-to-end time is proportional to the delay in being able to start task executions). If the agent does not have too much resources to expend on the negotiation or if there is a very slight probability that the other agent will accept the contract, then the local agent should choose the single shot protocol.

4. Failure of a negotiation to reach a commitment: If the negotiation between two agents using a particular negotiation protocol fails, the initiating agent should decide whether to retry the negotiation; whether to use the same protocol or an alternate protocol with the same agent or alternate agents and how many such retries should take place? **Heuristic Rule:** If negotiation is preferred (the agent will receive high utility as a result of the negotiation), then a more complex negotiation protocol is chosen since it has a higher probability of succeeding. Since resources have already been spent on figuring out a solution to the negotiation, it may be profitable to put in a little more effort and achieve a solution. If there is a very slight or no probability of finding an acceptable commitment, then resources which can be profitably spent on other solution paths are being wasted and the agent might find itself in a dead-end situation with no resources left for an alternate solution. So the negotiation option should be dropped.

Two other meta-level decisions which are being developed determine the parameters for invoking the domain scheduler including scheduler horizon, scheduler effort and slack amount in overall schedule and also determine whether to invoke the domain level scheduler for a reschedule since the performance of the agent is not going as expected.

Experimental Results

For the purposes of this paper, we used the environment introduced in the previous section with randomly generated which adheres to the above mentioned characteristics. The

Row #	Agent Name	TS ID	Arrival Time	Deadline	Control Activity	Utility						
						Run1	Run2	Run3	Run4	Run5	Run6	Run7
1	A	T0_1	1	40	NTCS	17.50	19.50	17.50	17.50	17.50	17.50	16.30
2	A	T0_2	10	28	Drop	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	A	T1_3	21	75	ATCS, NM1	45.60	51.19	55.20	45.60	45.60	12.00	43.41
4	A	T0_4	55	80	ATCS	16.00	16.00	16.00	16.00	16.00	16.00	18.00
5	A	T0_5	61	100	ATCS	6.00	6.00	6.00	6.00	6.00	6.00	6.00
6	B	NX_3	37	47	SS	10.00	10.00	10.00	10.00	10.00	-	10.00
Total Utility						96.10	102.69	104.70	95.10	95.10	51.50	93.77
Col. #	1	2	3	4	5	6	7	8	9	10	11	12

Table 1: Experimental Results for agents capable of meta-level reasoning: *A*gent Name is name of the agent being considered; *T*S ID is the name of the task being considered; *A*rrival Time and *D*eadline are the arrival times and deadlines for that particular task; *C*ontrol Activity is the control action chosen by the meta-level controller; Columns 5-11 describe the utility accrued for each of the individual tasks in seven different runs; Row 6 describes the total utility of all tasks completed by both agents for each run

Row #	Agent Name	TS ID	Arrival Time	Deadline	Control Activity	Utility						
						Run1	Run2	Run3	Run4	Run5	Run6	Run7
1	A	T0_1	1	40	ATCS	22.80	24.00	23.00	22.00	22.00	22.00	18.26
2	A	T0_2	10	28	ATCS	10.00	12.00	10.00	10.00	10.00	10.00	10.00
3	A	T1_3	21	75	ATCS, NM1	12.00	12.00	12.00	12.00	12.00	12.00	12.00
4	A	T0_4	55	80	ATCS	10.00	10.00	10.00	10.00	10.00	10.00	12.00
5	A	T0_5	61	100	ATCS	10.00	10.00	12.00	12.00	10.00	10.00	10.00
6	B	NX_3	39	53	ATCS	10.00	10.00	10.00	10.00	10.00	10.00	10.00
Total Utility						74.80	78.00	77.00	76.00	74.00	74.00	72.26
Col. #	1	2	3	4	5	6	7	8	9	10	11	12

Table 2: Experimental Results for agents with no meta-level reasoning: *C*ontrol Activity is the fixed control action used by the agent

maximum possible utility from task *T0* is 23.0 and minimum is 17.0; the maximum from task *T1* is 56.0 and minimum utility is 12.0; *NX* has a deterministic utility of 10.00. We make the simplifying assumption that task **NX** arrives at agent *B* only as a result of a successful negotiation with agent *A*. There are four possible meta-decisions upon arrival of a new task: **NTCS**, New Task Complex Scheduling invokes the complex DTC scheduler on the new task only and has a time cost of 2; **Drop**, this causes the agent to drop the new task and not reason about it ever again has a time cost of 0; **ATCS**, All Task Complex Scheduling invokes the complex DTC scheduler on the new task as well as all other tasks which are on the agenda or in partial execution and has a time cost of 3; and **SS**, Simple Scheduling invokes the simple abstraction based analysis on the new task only and has a time cost of 1. There are two possible options for Negotiation: **NM1**, Negotiation Mechanism 1 which is the simple single-shot protocol and **NM2**, Negotiation Mechanism 2 which is the more complex multi-shot protocol.

The design criteria in these experiments is to maximize overall utility over a finite horizon. Individual tasks have hard deadlines associated with them. It is assumed that if a task has not accrued utility by its deadline, it receives a utility of zero. This simple design criteria setting is one that lends itself to meta-level control as the existence of a hard deadlines (in contrast to a soft preference, e.g., soft deadline or no deadlines) make processor and other resources valuable commodities requiring a the non-myopic reasoning provided by the meta-level control component.

The results for the experiments on agents which have meta-reasoning capabilities are shown in Table 1 and the results on agents which have no meta-level reasoning capabilities are shown in Table 2. The above described scenario is used in both cases. All domain, control and meta-level actions have a time cost associated with them which are reflected in the results.

Consider Table 1 where each row represents a specific task arriving at the specified agent at the associated arrival time with a deadline. The task names are augmented with the arrival count to differentiate between various instances of the same task. For eg. Row 4 describes task **TO** arriving at agent *A* as its fourth task at time 55 with a deadline of 80. Column 5, titled *Control Action* describes the various decisions made by the meta-level controller upon arrival of the new task. Columns 6-12 describe the utility accumulated by each of the tasks for seven different runs.

In Row 1, task **T0_1** arrives at time 1. Since the meta-level controller is aware that no other tasks are in execution, it invokes **NTCS** on the task which is a cheaper option than **ATCS** which would be the choice of an agent with no meta-reasoning capabilities.

In Row 2, task **T0_2** arrives at time 11 while the previous task is still in execution and a meta-level decision to drop task **T0_2** is made. This is because the previous task **T0_1** has the exact same characteristics as the current task and has a tight deadline. The task also has a tight deadline and interrupting the already executing tasks might result in missing the deadlines on both Task **T0_1** and task **T0_2**.

In Row 3, Agent *A* decides to do a complete reschedule of all tasks and chooses to negotiate with agent *B* over task **NX** using negotiation mechanism **NM1**. In this case, it is willing to reschedule task **T0_1** since the expected utility from the newly arrived task is much higher than that of the current task. Also, the fact that the agent dropped task **T0_2** although it was unaware of the arrival of a highly preferred task in the near-future works to agent *A's* advantage since it has more time to perform the higher valued task. In five out of six runs, the agent's decision to drop the previous task **T0_2** and perform task **T1_3** with the negotiation option results in very high utility values. In Run 6, task **T1_3** receives a very low utility because negotiation fails with agent *B* and the task receives the minimum utility. However on average, agent *A*'s meta-level decision works to its benefit. In Row 6, we see that agent $\hat{B}$ chooses the simple scheduling option to execute task **NX_3** because of its tight deadline.

Consider Table 2. Here the agent does not reason about the characteristics of the tasks at the meta-level. This results in the agent choosing the same control action, namely **ATCS** for all tasks independent of the status of other tasks in execution. This results in the most expensive control action being invoked independent of the current state of the system. This results the choice of domain activities with shorter durations and lower utilities as reflected by the utility values in columns 6-12. The total utilities accumulated by five of the six runs in Table 2 is significantly lower than the corresponding run in Table 1. This supports our hypothesis that meta-level control is generally advantageous.

Conclusions and Future Work

In this paper we present a novel meta-level control agent framework for sophisticated multi-agent environments. The meta-level control has limited and bounded computational overhead and will support reasoning about scheduling and coordination costs as first-class objects.

We have shown, using a simple example, that meta-level control is beneficial. The heuristics described in this paper, although very simple, enable the meta-level controller to make accurate decisions in simple scenarios. We plan to introduce more complex features which will make the reasoning process more robust. Some such features include relation of slack fragments in local schedule to new task. This would enable an agent to fit a new task in its current schedule if it is possible and avoid a reschedule. Another feature would be to estimate the decommitment cost for a particular task. This will enable us to consider environments in which agents can decommit from tasks which they have previously agreed to complete.

We will extend the detailed domain level scheduler(DTC) to handle scheduling effort, slack and horizon as first-class objects. The extended DTC will accept parameters which constrain the effort spent on scheduling which in turn will affect the overhead of the scheduler. It will also be extended to deal with slack as a schedulable element which can be quantified and valued as any other primitive action. We hope that augmenting the domain level scheduler will make the meta-level controller more versatile by providing more options.

Finally, we will use the insight gathered from the heuristic approach to construct the state features, reward functions and algorithms to apply a reinforcement learning approach to this problem. We expect this analysis to provide valuable experience about applying RL techniques to complex real-world problems.

References

[1] Craig Boutlier. Sequential Optimality and Coordination in Multiagent Systems. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.

[2] Victor Lesser, Bryan Horling, Frank Klassner, Anita Raja, Thomas Wagner, and Shelley XQ. Zhang. BIG:an agent for resource-bounded information gathering and decision making. In *Artificial Intelligence Journal, Special Issue on INternet Applications*, 1999.

[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, Victor Lesser, and Thomas Wagner. Toward Robust Agent Control in Open Environments. In *Proceedings of the Fourth International Conference on Autonomous Agents*, pages 84–91, Barcelona, Catalonia, Spain, July,. ACM Press.

[5] H. Simon. From substantive to procedural rationality, 1976.

[6] Thomas Wagner, Alan Garvey, and Victor Lesser. Criteria-Directed Heuristic Task Scheduling. *International Journal of Approximate Reasoning, Special Issue on Scheduling*, 19(1-2):91–118, 1998. A version also available as UMASS CS TR-97-59.

[7] XiaoQin Zhang, Rodion Podorozhny, and Victor Lesser. Cooperative, multistep negotiation over a multi-dimensional utility function. In *IASTED International Conference, Artificial Intelligence and Soft Computing (ASC 2000), Banff,Canada*, pages 136–142. IASTED/ACTA Press, July 2000.