Controlling Deliberation in a Markov Decision Process-Based Agent

George Alexander
Department of Software and Information Systems
The University of North Carolina at Charlotte
Charlotte, NC
gralexan@uncc.edu

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

David J. Musliner
Honeywell Laboratories
Minneapolis, MN
david.musliner@honeywell.com

ABSTRACT

Meta-level control manages the allocation of limited resources to deliberative actions. This paper discusses efforts in adding meta-level control capabilities to a Markov Decision Process (MDP)-based scheduling agent. The agent's reasoning process involves continuous partial unrolling of the MDP state space and periodic reprioritization of the states to be expanded. The meta-level controller makes situation-specific decisions on when the agent should stop unrolling in order to derive a partial policy while bounding the costs of state reprioritization. The described approach uses performance profiling combined with multi-level strategies in its decision making. We present results showing the performance advantage of dynamic meta-level control for this complex agent.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Performance

Keywords

bounded rationality, Markov decision process, meta-level control

1. INTRODUCTION

Intelligent agents often must make effective use of limited resources (such as time, computational power, or physical resources) in order to achieve their goals. Deliberative activities such as planning, scheduling, and negotiating are used to manage these resources. However, an agent may have limited time to devote to deliberation and may not be able to reach globally optimal decisions in the time available. Thus it becomes important to maximize the effective use of limited deliberative resources. Meta-level control is the process of reasoning about and controlling the agent's deliberative actions. Examples of meta-level control questions are how to divide available deliberation time among the different deliberative actions available to the agent and what algorithms, parameters, and protocols the agent should use for deliberation if multiple options are available.

This paper describes efforts to add meta-level control capabilities to the IU-agent [14, 19], a scheduling agent based on the Markov Decision Process (MDP) formalism designed to operate in a cooperative multi-agent environment. Although the IU-agent can perform deliberative actions and domain actions simultaneously, the agent's tasks involve temporal constraints that necessitate intelligent management of deliberation. The goals of the research are to implement a meta-level control scheme to maximize the effectiveness of the IU-agent's deliberations (namely, determining when the agent should update its MDP policy and taking measures to ensure the agent stays on policy for as long as possible) and to identify characteristics of domains in which meta-level control of the IU-agent proved advantageous.

The rest of the paper is laid out as follows: first is a brief discussion of related work in meta-level control, especially in the area of performance profiling. Background information is provided on Markov Decision Processes and the TAEMS modeling language (a derivative of which was used to represent the IU-agent's tasks) along with a description of the IU-agent. Next is a description of the implemented meta-level control approach followed by experimental results indicating its advantages for the IU-agent. The paper is concluded with a discussion of lessons learned over the course of implementing meta-level control for the IU-agent and an indication of a number of open issues for future work in meta-level control.

Related Work

There has been important previous work in meta-level control [5]. Russell and Wefald [17] describe an expected utility-based approach to decide whether to continue deliberation or to stop it and choose the current best external action. They introduce myopic schemes such as meta-greedy algorithms, single step and other adaptive assumptions to bound the analysis of computations. In [16], Raja and Lesser present a decision-theoretic approach that leverages an abstract representation of the agent's state to bound the cost of meta-level decision making in a complex multi-agent environment.

Boddy and Dean [3] describe a decision-theoretic approach to scheduling deliberative actions implemented by anytime algorithms [6]. These algorithms are guaranteed to return a

Cite as: Controlling Deliberation in a Markov Decision Process-Based Agent, George Alexander, Anita Raja, and David Musliner, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. XXX-XXX.

result at any time they are interrupted, and it is assumed that their solution quality increases as they are given more computational time. Their approach uses performance profiles, which provide a measure of how the solution quality changes over time. Hansen and Zilberstein [9] present a formal approach to meta-level control of anytime algorithms that reasons explicitly about monitoring costs. More recently, Larson and Sandholm [11] evaluate the use of performance profile trees, which stochastically model the *path* the solution takes. Online meta-level control of an adaptive real-time agent is investigated in [8,15]. The meta-level control scheme described in this paper uses the simplified representation of a performance profile as a curve and adds a multi-leveled decision making strategy. The performance profiles are not assumed a priori, but are induced experimentally and maintained through periodic updates.

2. BACKGROUND

2.1 Markov Decision Processes

A Markov Decision Process is a probabilistic model of a sequential decision problem, where states can be perceived exactly, and the current state and action selected determine a probability distribution on future states [18]. Specifically, the outcome of applying an action to a state depends only on the current action and state (and not on preceding actions or states). Formally, an MDP is defined via the 4-tuple $\langle \mathcal{S},\ \mathcal{A},\ \mathcal{P},\ \mathcal{R} \rangle$: a state set $\mathcal{S}$, an action set $\mathcal{A}$, a transition probability function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0,1]$, and a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \Re$. On executing action a in state s, the probability of transitioning to state s' is denoted $P^a_{ss'}$ and the expected reward associated with that transition is denoted $R^a_{ss'}$. A rule for choosing actions is called a *policy*. Formally, it is a mapping $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0,1]$ (if the policy is deterministic, we may simplify this as $\pi : \mathcal{S} \mapsto \mathcal{A}$). If an agent follows a fixed policy, then over many trials, it will receive an average total reward known as the *value* of the policy. In addition to computing the value of a policy averaged over all trials, we can also compute the value of a policy when it is executed starting in a particular state s. This is denoted $V^\pi(s)$ and it is the expected cumulative reward of executing policy π starting in state s. This can be written as

$$V^\pi(s) = E[r_{t+1} + r_{t+2} ... | s_t = s, \pi]$$

where r_t is the reward received at time t, s_t is the state at time t, and the expectation is taken over the stochastic results of the agent's actions.

For any MDP, there exists one or more optimal policies which we will denote by π^* that maximize the expected value of the policy. All of these policies share the same optimal value function, written as V^* The optimal value function satisfies the Bellman equations [2]:

$$V^*(s) = \max_a \Sigma_{s'}\ P(s'\ |s,a)[R(s'\ |s,a) + V^*(s'\ )]$$

where $V^*(s')$ is the value of the resulting state s'.

2.2 TAEMS Modeling Language

TAEMS models [10] are hierarchical abstractions of multi-agent problem solving processes that describe alternative ways of accomplishing a desired goal; they represent major problems, decision points, and interactions between problems. Non-leaf nodes in the hierarchy are called *tasks* (tasks may have one or more subtasks), while nodes at the lowest level are called *methods* and represent the domain-level actions available to an agent. Methods are characterized by discrete probability distributions along three dimensions: quality, duration, and cost. Quality Accumulation Functions (QAFs) define how quality propagates from a child node to its parent. Example QAFs are *q_sum*, indicating the parent is assigned the sum of qualities from its children, *q_max*, indicating the parent is assigned the maximum quality of any of its children, and *q_min*, indicating the parent is assigned the minimum quality of any of its children. In addition, temporal constraints may be represented by assigning earliest start times and deadlines to nodes; these constraints limit the time period available to achieve quality for the node. Temporal constraints propagate downward through the hierarchy. Finally, task interrelationships may be represented by linking nodes using non-local effects (NLEs). Example NLEs include hard constraints, such as *enables* (the enabled tasks cannot accrue quality until the enabling task has achieved non-zero quality), as well as soft constraints, such as *facilitates* (the facilitated task will accrue more quality if it is executed after the facilitating task has achieved non-zero quality). An example TAEMS structure is given in Figure 1.

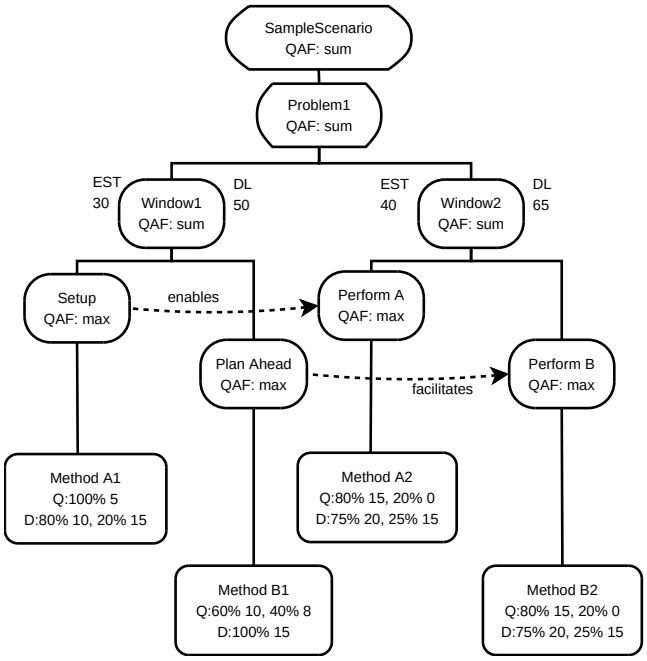


Figure 1: An example TAEMS structure.

A TAEMS model can be converted into an equivalent MDP and solved to obtain an optimal policy [19]. In fact, due to the deadline constraints in the TAEMS model, the conversion will produce a finite horizon MDP. Starting from the initial state, successor states are generated based on all the actions available to the agent in a state and all possible outcomes of those actions. For example, if the agent could choose between two actions and these actions had no uncertainty in quality, cost, or duration, then two successor states would be generated. In general, if n actions are available in a

given state, each having q possible quality values, c possible costs, and d possible duration values, then $n \cdot q \cdot c \cdot d$ immediate successor states would be generated. In practice, even for fairly small TAEMS models, if there is a lot of flexibility in action choices and many possible outcomes of each action, the MDP may be too large for the agent to completely enumerate in time. The IU-agent thus performs this conversion incrementally and continually, alongside domain-level task execution. Although the IU-agent derives its MDP from a TAEMS model, our meta-level control approach does not depend on TAEMS per se, just the incremental unrolling process.

2.3 IU-agent

The IU-agent [19] translates its task model into an MDP beginning with the earliest states and expanding toward the problem horizon. This unrolling process must be periodically paused in order to derive a partial policy based on the currently expanded states and continues until either the entire MDP is unrolled, the scenario deadline is reached, or the agent arrives in an unknown state and cannot remain on policy. States that have not yet been expanded are maintained in a queue called the open list. If unrolling proceeds naively in a breadth-first manner, then the agent may only be able to expand states representing the near future and will consequently perform poorly. If the agent cannot unroll enough of the state space, or more specifically, a long enough view of the state space, the resulting policy may be sub-optimal. As more states are unrolled and the (partial) MDP becomes more forward-looking, the agent's partial policy approaches the optimal policy for the complete MDP. Thus it is natural to try to discover heuristics that guide the unrolling process in a way that maximizes the expected quality of the agent's policy given limited computational time.

The process of unrolling the MDP without such heuristic guidance is called *uninformed unrolling*. In contrast, the IU-agent uses a method called *informed unrolling* [19], from which it derives its name. Informed unrolling is based on the principle that the agent should not spend much time unrolling states which will probably never be reached; thus the IU-agent prioritizes unexpanded states in the open list according to the probability of reaching those states when following an optimal policy. Policy execution occurs alongside unrolling; hence as the actual outcome of actions becomes known, the probability of reaching a state changes and some states become unreachable. Therefore the IU-agent periodically removes or "prunes" unreachable states from the MDP and the open list.[1] Pruning may be triggered by the agent making an action choice (eliminating the states representing the action(s) not chosen) or by the completion of a method (eliminating the states representing those possible outcomes that did not in fact occur). In addition, the open list is periodically sorted so that states with the highest probability of being reached are placed toward the front of the queue. After sorting the open list, the IU-agent solves the current MDP to obtain a new partial policy. Meanwhile, the unrolling process remains paused until the sorting/policy derivation is complete. Policy derivation is costly [12]; thus meta-level control is needed to balance the time spent unrolling vs. the time spent sorting the open list and deriving a policy. The meta-level controller initiates the process by instructing the agent to perform an open list sort, which in turn triggers the policy derivation function. Since this sequence is opaque to the meta-level controller, we will refer to the entire procedure as a "deliberative action" that is triggered by an open list sort. Also, if a possible prune is detected while the agent is in the middle of the sorting process, the sort will be aborted and the prune processed, then a new sort is begun using the pruned open list. Obviously, interrupting many sorts over the course of the agent's execution becomes expensive, wasting the computational time already spent before the sorts are aborted. Therefore, the agent requires adaptive control to schedule open list sorts at the most appropriate times.

Since the unrolling process occurs alongside execution of domain actions, the agent may enter an unknown state or a state not considered in the agent's current partial policy. In this event, the agent abandons its MDP-based reasoning for a myopic greedy action selection method.

[1]Only states that are intrinsically unreachable are removed (e.g., states corresponding to an earlier time period) States which could not be reached by the current policy are retained, since the policy may change as more of the state space is explored.

3. META-LEVEL CONTROL APPROACH

The meta-level control component for the IU-agent is designed with a number of goals in mind. The most basic motivation is the trade-off between keeping the agent's MDP policy current (by taking frequent deliberative actions) and maintaining a more forward-looking MDP (by unrolling more states). In addition to this consideration, it is desirable to reduce the amount of deliberation time wasted by the agent on sorts that are interrupted by prunes before completion, since these sorts-in-progress are thrown out and the wasted time could have been spent unrolling more MDP states. The third goal is to ensure that the agent stays on policy for as long as possible, rather than fall back on a greedy myopic deliberation method.

The agent can fall off policy in two different ways: "off the end" or "off the side." When the agent reaches an area of the state space that has not yet been unrolled, the agent has fallen off the end of its policy. If the agent reaches a state that has been unrolled but not yet incorporated into its policy (that is, no action has been associated with the state), it has fallen off the side of the policy. Falling off the end of the policy can be avoided by maximizing the number of MDP states unrolled, for example by reducing aborted sorts and limiting the amount of time spent in deliberation. On the other hand, falling off the side of the policy can be avoided by maintaining an up-to-date policy; that is, by increasing the frequency of deliberative actions. Thus the meta-level control component has to balance competing goals.

The meta-level control approach consists of building performance profiles for the agent's deliberative action and using a number of heuristics to determine whether to initiate a sort (triggering deliberation) or continue unrolling, discussed in the following subsections. The general algorithm is summarized in Procedure 1. Each time a state is expanded from the open list, the IU-agent makes a decision about whether to pause unrolling, resort the open list, and derive a new partial policy, or just to continue unrolling. A single primary heuristic is tested first. If the primary heuristic does not return *true*, then one or more secondary heuristics are tested sequentially. If none of the heuristics return *true*, then the IU-agent continues unrolling the MDP.

Procedure 1 Meta-level Control Loop

```
1: loop
2:    expand a state from the open list
3:    if primaryHeuristic then
4:       call openList sort and derive policy
5:    else if secondaryHeuristic_1 then
6:       call openList sort and derive policy
7:                    ⋮
8:    else if secondaryHeuristic_N then
9:       call openList sort and derive policy
10:   end if
11: end loop
```

3.1 Performance Profiles

Initial data confirmed that the time required to reorder the open list as well as the time required for policy derivation (collectively, the *deliberation time*) scales with the number of reachable states in the agent's MDP; thus we collected from the agent's logs the deliberation time and reachable state count for all the deliberative actions taken across a number of domains and used this data with gnuplot's [7] built-in curve-fitting feature (based on the Marquardt-Levenburg algorithm [13]) to create a deliberation time estimator function. For this research, the correlation between reachable states and deliberation time was treated as domain independent (in other words, the performance of open list sorting and policy derivation depends just on the number of reachable states and not the particular topology of the MDP). Example deliberation time data is shown in Figure 2 with the estimation function overlaid. Notice that the data follow a roughly linear trend until about 60,000 reachable states, when the sort times sharply increase. To account for this, the data are fitted with the following piece-wise function:

$$f(x) = \begin{cases} ax + b & if \quad x < 60000 \\ cx^3 + dx^2 + ex + f & if \quad x \geq 60000 \end{cases}$$

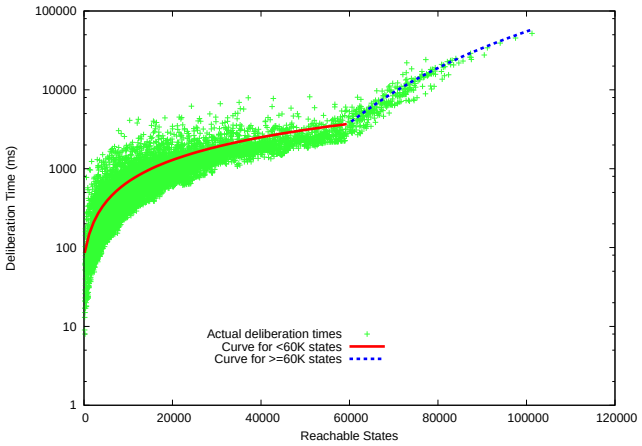


Figure 2: Deliberation time data before memory optimizations. Note the sharp increase in deliberation times after approx. 60K reachable states.

Investigations suggested that the sharp increase in deliberation times was caused by garbage collection issues. After some memory optimizations were performed on the IU-agent's unrolling algorithm, the data followed a smoother curve. Timing data collected after these modifications were made is shown in Figure 3, along with a re-tuned estimation function. The curve-fitting procedure had to be periodically re-run using the latest data as other changes were made to the agent; however, the overall trend of the data remained the same. These events hint at the possible interactions even at the coding level between meta-level control and deliberative control.

Note that the true number of reachable states is unknown to the agent until after policy derivation is completed. For input to the deliberation time function, this number was estimated by the sum of the previous reachable state count, the number of states added to the open list, and the number of true terminal states added to the MDP.[2]

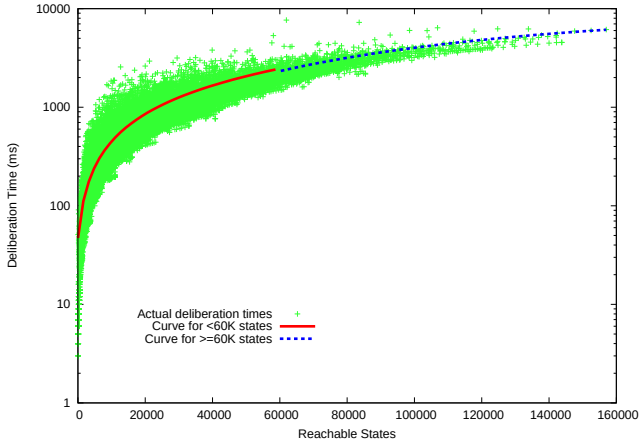


Figure 3: After memory optimizations, the trend in deliberation time data is much smoother.

3.2 Time-to-sort Heuristics

The estimated deliberation time was used as input to several heuristic functions that determine whether the agent should stop unrolling and trigger an open list sort at a given point in time. These heuristics are divided into two categories: *primary* heuristics and *secondary* heuristics. Primary heuristics represent rules for when to initiate a sort under normal conditions. They are responsible for the periodic policy derivation that is necessary without any special considerations. Secondary heuristics represent rules for sorting in certain special cases; they allow the agent to opportunistically react when deriving an updated MDP policy is particularly desirable.

3.2.1 Primary Heuristics

The IU-agent uses a single primary heuristic which does not change over the course of a scenario. We considered a number of possible heuristics, described below; however, our experiments suggested that the *sort-budget* rule performed best in the domains of interest. We hypothesized that the best choice of primary heuristic may depend on characteristics of the domain, but no clear trends in this regard were discovered over the course of the experiments.

[2]true terminal states are actual terminal states of the complete MDP, distinguished from states which are merely at the edge of the partially-unrolled MDP.

n-pops-and-growth This rule triggers an openlist sort with associated policy derivation to be performed whenever N states have been popped off the openlist and expanded and the MDP has grown by M states. Formally,

$popped \leftarrow$ # of states popped off open list
$growth \leftarrow$ # of states added to MDP
if $(popped \geq N) \wedge (growth \geq M)$ **then**
  return true
end if

One of the challenges in effectively applying this rule is deciding on appropriate values of N and M such that sorting occurs regularly but not too frequently.

process-openlist-n-percent This rule is invoked whenever N% of the states from the openlist have been expanded. Intuitively, the rule causes more frequent sorting (thus a more focused MDP) during the beginning of a scenario with less frequent sorting (thus more breadth-first exploration) after the MDP becomes larger. Formally,

$prevSize \leftarrow$ # of open list states after last sort
$popped \leftarrow$ # of states popped off open list
if $popped \geq \frac{N}{100} \cdot prevSize$ **then**
  return true
end if

sort-budget This rule times sorts so that roughly a certain fraction of the agent's time is spent on deliberative actions versus unrolling. Let b be the budget amount (i.e., the fraction of the time relative to unrolling time that we wish to spend deliberating), then formally,

$unrollTime \leftarrow$ time since last sort
$delibDuration \leftarrow$ estimated deliberation time
$deliberationRatio \leftarrow delibDuration/unrollTime$
if $deliberationRatio \leq b$ **then**
  return true
end if

In order to prevent time lost to aborted sorts, these primary heuristics are constrained by an additional condition (given in Procedure 2) that the triggered deliberative action will not run over a method start time or possible method completion time (which would cause some states to be marked unreachable, hence aborting the sort-in-progress when these states are subsequently pruned from the MDP). A sort is triggered by the primary heuristic returning *true* only if Procedure 2 returns *true* as well.

Procedure 2 not-too-close-to-next-possible-prune

$delibDuration \leftarrow$ estimated deliberation time
$nextEST \leftarrow$ next earliest start of a method
$nextFinish \leftarrow$ next possible completion time of the currently executing method
$nextPossPrune \leftarrow min(nextEST, nextFinish)$
if $currentTime + delibDuration \leq nextPossPrune$ **then**
  return true
end if

3.2.2 Secondary Heuristics

In addition to the primary heuristic used to determine when to perform routine openlist sorting/policy derivation, the IU-agent uses multiple secondary heuristics to decide whether to take an opportunistic deliberative action should certain conditions occur.

perfect-time-to-sort This condition is triggered when the next deliberation is estimated to complete within a small time window before the IU-agent's next action choice time. The purpose of this rule is to try to ensure that the MDP policy is always as up-to-date as possible whenever a domain-level decision is made. Formally,

$\Delta \leftarrow$ the size of the desired time window
$delibDuration \leftarrow$ estimated deliberation time
$timeAvail \leftarrow$ time left until next earliest start of a method
if $0 \leq timeAvail - delibDuration \leq \Delta$ **then**
  return true
end if

need-actions-for-near-term-states This heuristic examines the reachable non-edge MDP states that correspond to times within a small window of the agent's current time. If any of these states do not have actions associated with them in the MDP policy, the rule triggers an open list sort. The reasoning behind this rule is that if the agent enters a state for which the policy has no associated action, then the agent will fall off the policy and use a myopic deliberative method for the duration of the scenario. Let π be the agent's MDP policy, with $\pi(s)$ denoting the action associated with state s, and let $time_s$ denote the time of state s. Furthermore, let Δ be the size of our desired time window, and let $NearTerm = \{s \in ReachableStates : time_s \leq currentTime + \Delta\}$. Then this rule is given formally as

for all $s \in NearTerm$ **do**
  if $s \notin EdgeStates \wedge \pi(s) == NULL$ **then**
    return true
  end if
end for

These secondary heuristics may trigger a sort at a time when it would normally be interrupted by a prune; however, since in the IU-agent's domain of interest the special conditions represented by these rules are considered more important than the benefits of processing prunes, sorts triggered by secondary heuristics are programatically defined to be uninterruptible.

4. EXPERIMENTAL RESULTS

The meta-level control approach was tested on a total of 169 domains, divided among 8 groups. The domains in Groups 1–3 and 7–8 were designed by external teams. Unfortunately, due to the size and complexity of the TAEMS models, it is difficult to give a succint general description of these domains without knowledge of the motivations behind their design. However, the domains in Groups 4–6 were designed specifically to capitalize on the strengths of the meta-level control approach, focusing on two of the main intended benefits to the agent: increased number of unrolled states

Group	Mean Qual. (MLC Off)	Mean Qual. (MLC On)	N	Sig. (1-tail paired t)
1	1188.82	1185.74	25	0.147
2	29.74	41.37	25	0.218
3	18.73	15.23	25	0.275
4	573.75	613.91	25	0.001
5	0.00	2.33	18	< 0.001
6	155.56	178.17	7	0.079
7	6.44	6.44	25	——
8	184.85	463.04	19	0.066
All	297.73	335.23	169	0.030

Table 1: Mean quality comparison

to avoid falling off the end of the policy and monitoring to prevent falling off the side of the policy. These domains are summarized as follows:

Group 4 : An assortment of domains, mostly consisting of small chains of enabling NLEs along with scattered single NLEs. The design strategy is to give the agent many action choices and to require the agent to reason non-myopically.

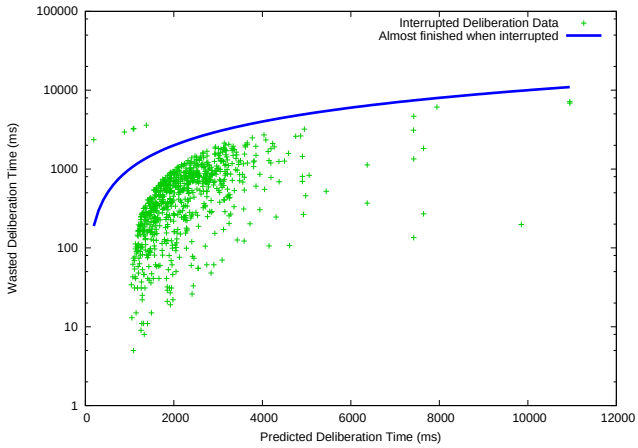
Group 5 : Small domains consisting of long chains of enabling NLEs. Quality in these domains is essentially binary. If the agent cannot unroll the complete state space in time, then it will achieve 0 quality; otherwise, the agent gets a small amount of quality.

Group 6 : Domains containing sets of low quality actions that enable high quality actions, with a single very high quality action. The agent has to maximize the number of unrolled states in order to see the very high quality state.

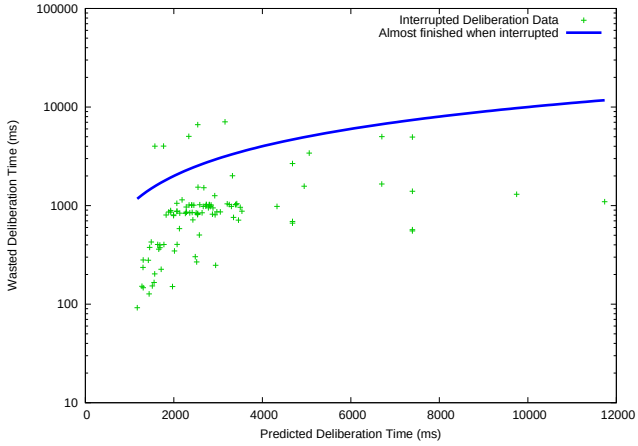
The results (see Table 1) show a statistically significant increase in mean quality achieved for the IU-agent with meta-level control in Groups 4–5 in particular and for all groups regarded as a whole (overall mean quality increase $\approx 12.6\%$). On many of the individual domains comprising Groups 2, 3, and 7, the agent was not able to achieve quality with or without meta-level control, and further investigation suggested that these domains may require communication capabilities more advanced than those implemented in the IU-agent at the time of the experiments. However, log information for runs of Groups 2 and 3 reveals that the agent *w*as able to stay on policy longer with meta-level control enabled (Figures 4(a) and 4(b) respectively).

Additionally, meta-level control was able to greatly reduce the number of open list sorts that were aborted and restarted. Figure 5 shows the time wasted on aborted sorts over a run of all the domains in Group 4. The line $y = x$ is shown for illustration: points near the line indicate that the deliberation was almost completed when it was interrupted. With meta-level control enabled, the IU-agent had about 12.7% of the aborted sorts of the agent lacking meta-level control.

5. DISCUSSION



(a) Meta-level control disabled



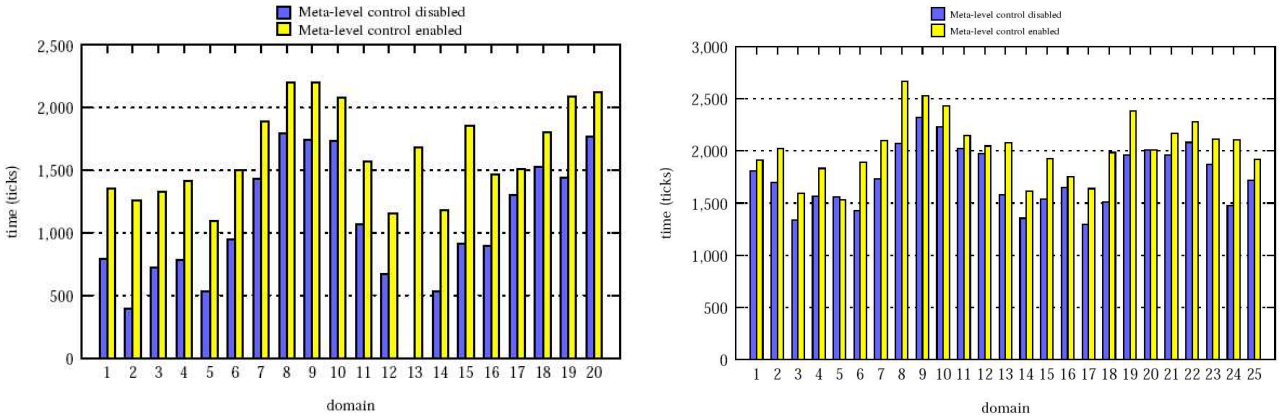
(b) Meta-level control enabled

l

Figure 5: Time wasted on interrupted sorts (Group 4 domains). With the meta-level control features disabled, the agent estimated deliberation time by multiplying the time spent on the previous deliberative action by a constant ratio.

Meta-level control is not a panacea [4]. In our experiments, we found several situations in which the agent performed just as well with meta-level control disabled. However, certain domain characteristics favored sophisticated meta-level control:

- High amounts of uncertainty or many options for domain actions require the IU-agent to unroll its MDP in a more breadth-first manner in order to avoid falling off the side of the policy (that is, if there are many (almost) equally likely outcomes for an action choice, then the agent must unroll several subtrees instead of focusing on a single high-probability subtree). Meta-level control's advantage in this case is in the increased number of states unrolled compared with the meta-level control-disabled IU-agent.
- Non-myopic decision making should be required, for example long chains of *enables* NLEs or time constraints that require smart scheduling (for example,



(a) Group 2 domains

(b) Group 3 domains

Figure 4: Time spent on policy. Meta-level control allows the agent to remain on policy longer.

actions with short time windows between their release times and deadlines). If the IU-agent's myopic fallback deliberation performs well in a domain, then much of the advantage of MDP-based deliberation would be lost.

Conversely, certain domain characteristics reduce or eliminate the advantages of meta-level control for the IU-agent:

- In domains that are very loosely constrained (for example, no non-local effects, plenty of slack in the agent's schedule, etc.), a greedy approach may perform well. If such a ceiling effect eliminates the advantages of the IU-agent's reasoning capabilities, then of course meta-level control would not give it much added advantage.
- In domains requiring capabilities not present in the IU-agent, the agent may have a difficult time achieving any quality even with meta-level control. This appears to be the case for Groups 2, 3, and 7 in our experiments.

In short, the described meta-level control scheme may be viewed as enhancing the existing reasoning capabilities of the IU-agent; thus it should be beneficial in domains that take particular advantage of those abilities and not as effective in domains that do not leverage the agent's abilities.

One major lesson learned over the course of our research is that meta-level control should be considered from the beginning and developed alongside deliberation, since they affect each other. Changes in deliberation can require meta-level control changes; for example, on several occasions we noticed that the IU-agent performed as well without meta-level control on certain domains where meta-level control had formerly proven advantageous, and we had to adjust the parameters of our meta-level control to regain the advantage. Conversely, observations made while implementing the meta-level control strategy, such as gathering performance profile data, suggested areas for improving the deliberation of the agent.

6. CONCLUSION AND FUTURE WORK

We have described an approach to controlling deliberation in an MDP-based scheduling agent. The approach consists of collecting performance profile information from the agent empirically and using these profiles in a multi-level heuristic decision-making strategy. Experimental results were presented that suggest the advantages of meta-level control for the agent in a subset of test domains.

There are many open opportunities in meta-level control, especially in multi-agent systems [1]. In particular, we are working on how to extend isolated meta-level control of individual agents to distributed meta-level control of groups of cooperative agents. This involves many of the same issues as *deliberative control* of multiple agents, such as the need for coordination and negotiation.

7. ACKNOWLEDGEMENTS

We would like to acknowledge Ed Durfee's contribution to the need-actions-for-near-term-states heuristic. We also thank the three anonymous reviewers for their helpful comments. This material is based upon work supported by the DARPA/IPTO COORDINATORs program and the Air Force Research Laboratory under Contract No. FA8750–05–C–0030. The views and conclusions contained in this document are those of the authors, and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

8. REFERENCES

[1] G. Alexander, A. Raja, E. H. Durfee, and D. J. Musliner. Design paradigms for meta-control in multi-agent systems. In *Proceedings of the AAMAS 2007 Workshop on Metareasoning in Agent-based Systems*, pages 92–103, Honolulu, HI, May 2007.

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

[3] M. Boddy and T. Dean. Decision-theoretic deliberation scheduling for problem solving in time-constrained environments, 1994.

[4] V. Conitzer and T. Sandholm. Definition and complexity of some basic metareasoning problems. In *IJCAI*, pages 1099–1106, 2003.

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

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

[7] Gnuplot. http://www.gnuplot.info.

[8] R. P. Goldman, D. J. Musliner, and K. D. Krebsbach. Managing online self-adaptation in real-time environments. In *Lecture Notes in Computer Science*, volume 2614, pages 6–23. Springer-Verlag, 2003.

[9] E. A. Hansen and S. Zilberstein. Monitoring and control of anytime algorithms: A dynamic programming approach. *Artif. Intell.*, 126(1-2):139–157, 2001.

[10] B. Horling, V. Lesser, R. Vincent, T. Wagner, A. Raja, S. Zhang, K. Decker, and A. Garvey. The TAEMS White Paper, January 1999.

[11] K. Larson and T. Sandholm. Using performance profile trees to improve deliberation control. In *AAAI*, pages 73–79, 2004.

[12] M. L. Littman, T. Dean, and L. P. Kaelbling. On the complexity of solving markov decision problems. In *UAI*, pages 394–402, Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI–95).

[13] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, June 1963.

[14] D. J. Musliner, E. H. Durfee, J. Wu, D. A. Dolgov, R. P. Goldman, and M. S. Boddy. Coordinated plan management using multiagent MDPs. In *Working Notes of the AAAI 2006 Spring Symposium on Distributed Plan and Schedule Management*, pages 73–80, March 2006.

[15] D. J. Musliner, R. P. Goldman, and K. D. Krebsbach. Deliberation scheduling strategies for adaptive mission planning in real-time environments. In *Proc. Third International Workshop on Self Adaptive Software*, 2003.

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

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

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

[19] J. Wu and E. H. Durfee. Solving large TAEMS problems efficiently by selective exploration and decomposition. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 07)*, pages 291–298, Honolulu, HI, May 2007.