Design Paradigms for Meta-Control in Multiagent Systems

George Alexander[1], Anita Raja[1], Edmund H. Durfee[2], and David J. Musliner[3]

[1] Department of Software and Information Systems, The University of North Carolina at Charlotte, Charlotte, NC, {gralexan, anraja}@uncc.edu
[2] EECS Department, University of Michigan, Ann Arbor, MI, durfee@umich.edu
[3] Honeywell Laboratories, Minneapolis, MN, David.Musliner@honeywell.com

Abstract. In this paper, we attempt to define a generalized framework for meta-level control in multiagent systems. We generalize and extend previous work in single-agent meta-control. We discuss the issues which system designers must consider when designing an agent's meta-control component and conclude with areas for future research.

1 Introduction

1.1 What Is Meta-Control?

In a multiagent system, intelligent agents can be thought of as having at least three broad categories of available actions (see Fig.1): domain actions (such as movement), deliberative actions (such as scheduling and coordination), and meta-level control [1–4] actions. Domain actions cause changes to the agent's environment, and the agent uses deliberative actions to decide which domain actions to perform and how and in what order to perform them. The meta-level control layer makes similar decisions regarding deliberative actions. Thus the meta-level can be thought of as an abstraction of the deliberative layer.

We begin this paper with a discussion of the need for meta-control and a review of previous work in the area. We then describe a generalized single-agent meta-control framework and discuss some of the design issues involved in implementing our approach. Next, we discuss ways in which our single-agent framework could be extended to handle multiple cooperative agents which may need coordinated meta-control. We describe some of the complications that arise in trying to develop multiagent meta-control. Finally, we conclude with directions for further research.

1.2 Why Is Meta-Control Needed?

Open environments are dynamic and uncertain. It is paramount for complex agents operating in these environments to adapt to the dynamics and constraints of such environments. The agents have to deliberate about their local problem solving actions and coordinate with other agents to complete problems requiring

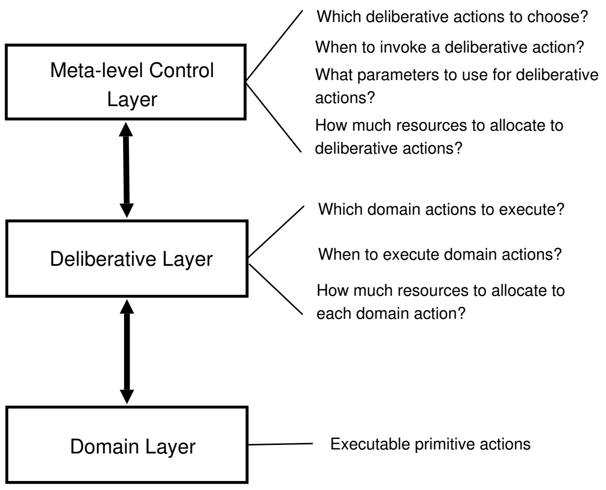
Meta-level Control Layer
Which deliberative actions to choose?
When to invoke a deliberative action?
What parameters to use for deliberative actions?
How much resources to allocate to deliberative actions?
Deliberative Layer
Which domain actions to execute?
When to execute domain actions?
How much resources to allocate to each domain action?
Domain Layer
Executable primitive actions

Fig. 1. Three categories of agent actions.

joint effort. These deliberations have to be performed in the context of bounded resources, uncertainty of outcome and incomplete knowledge about the environment. Furthermore, new problems with deadlines can be generated by existing or new agents at any time.

These factors pose a number of challenges for meta-level control systems. The agent may have uncertainties of outcome at both the domain layer and the deliberative layer. Also, the agent may need to choose between multiple possible deliberative actions. For example, scheduling versus coordinating with other agents, or using extra time for learning activities. The agent must be able to make intelligent tradeoffs in computation time versus quality of results. When coordinating with others, the agent must also consider whether communication costs are justified (c.f. [5]). Also, multiple agents may need to coordinate their meta-control activities. And this must all happen in the context of a dynamic environment using limited knowledge and resources! In fact, in the kind of complex, stochastic environments that agents may have to deal with, the meta-level control challenge goes beyond the traditional issue in which the uncertainty is in whether devoting more time to a deliberation will yield a commensurately better domain-level decision. It goes beyond because in such environments, not only is there uncertainty about how much better the decision will be, but also whether the decision will be needed at all given that the execution trajectory might veer such that any particular domain-level decision about future action is obviated. This suggests, then, that we need a *policy* for deliberation, where we consider the potential value of alternative deliberative activities given possible future state trajectories (where state reflects outcome of previous deliberation, as well as possibly of domain-level actions).

1.3 Related Work

There has been important previous work in meta-level control (for a review of metacognition in general, including metacognition in problem solving and meta-level control, see [6]). Russell and Wefald [4] 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. Schut and Wooldridge [7] have independently observed that a Markov Decision Process-based model towards decision making is most similar to the bounded optimality model. Russell, Subramanian, and Parr [8] cast the problem of creating resource-bounded rational agents as a search for the best program that an agent can execute. In searching the space of programs, the agents, called bounded-optimal agents, can be optimal for a given class of programs or they can approach optimal performance with learning, again given a limited class of possible programs. Our approach to meta-level control involves construction of agents similar to these bounded optimal agents. We too do not assume complete accessibility to the environment, which makes our approach applicable to a wide range of problems and delivers an execution model which makes it relevant to real-world applications. While our model has targeted only finite horizon problems, it accounts for computational resources and takes advantage of the Markov Decision model to bound computation and handles multiple inter-dependent meta-level questions. This work extends the meta-level control architecture described in the context of multiagent systems [3]. However there is a crucial difference. The MetaMod framework in this paper does not make any assumptions about a priori knowledge of the performance characteristics of the different deliberation actions on problems. The problem classifier component makes real time predictions about these performance characteristics based on actual performance of previously seen training problems.

2 A Single-Agent Meta-Control Framework

Meta-level control actions are essentially an abstraction of deliberative actions. That is, meta-level reasoning about deliberative actions follows many of the same principles and shares many of the constraints as deliberative reasoning about domain actions. Additionally, decisions made at the meta-level affect which deliberative actions are executed which in turn affect which domain actions are executed; thus meta-control affects the domain layer in a high-level way. Therefore it makes sense that the task presented to the agent's meta-control component be an abstract view of the agent's tasks at the lower levels. As mentioned previously, many times an agent's environment is too complex and uncertain for myopic reasoning. The agent needs to formulate an end-to-end plan for meta-control, or at least as non-myopic a plan as is computationally feasible, which takes into account the various possible outcomes of its decisions and their affect on future decisions in order to maximize overall utility. One formalism with the desired properties is the Markov Decision Process (MDP) [9]. These ideas of problem abstraction and sequential decision making are considered in our single-agent meta-control framework described below.

Figure 2 gives a high-level view of the control flow within our meta-control framework for a single agent (more detailed explanations of each component are given in the subsections following). When the meta-control component is triggered, perhaps by the arrival of a new high-level problem for the agent, the *Problem Abstraction Component* generates an abstract meta-level problem based on the agent's current problem solving context and available deliberative actions. Information about the expected results of each deliberative actions is gathered from a database of performance profiles (Steps 1-3). The *Decision Process Component* transforms this abstract problem into an MDP, which is then solved to obtain a non-myopic meta-level control policy. Based on this policy and the agent's current state, the Decision Process Component chooses the appropriate deliberative action (Steps 4-6). Finally, the deliberative action is executed, possibly resulting in further domain-level actions, and the performance results are used to update the performance profile database (Steps 7-9).

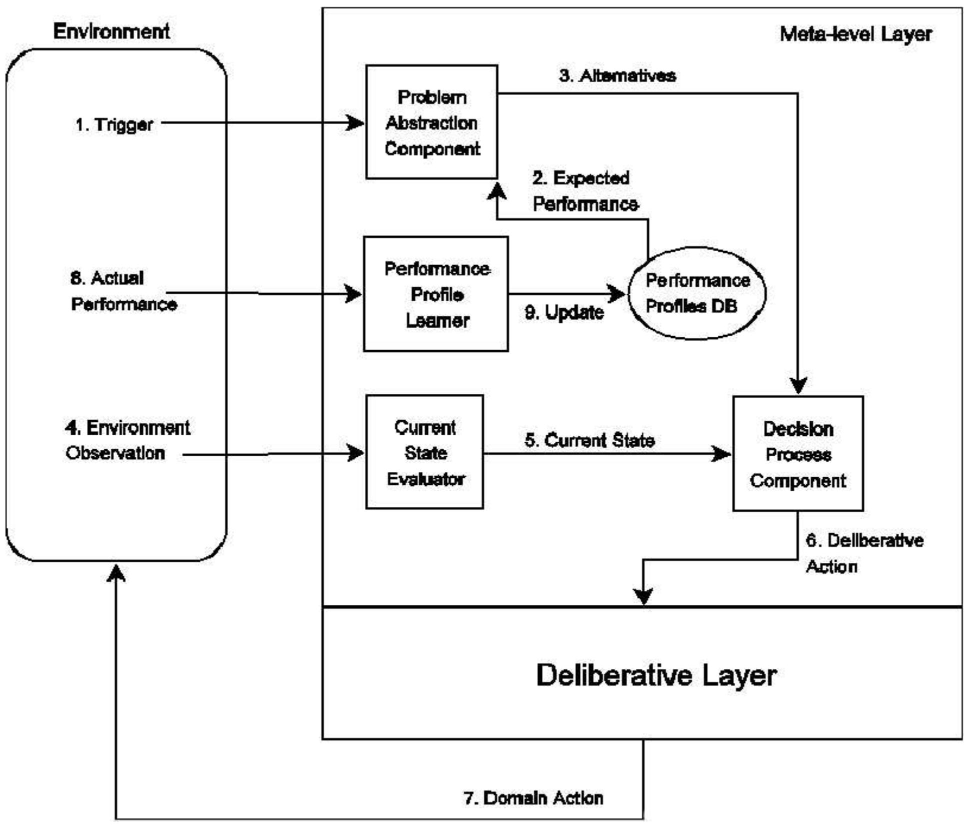

Fig. 2. Single-agent meta-control flow diagram.

2.1 Problem Abstraction Component

In this section, we give a more detailed description of the reasoning within the Problem Abstraction Component. From the agent's current problem solving context we derive problem features which are used to classify the context into one of several performance types (categories based on the predicted performance of various deliberative actions). Each performance type has an associated performance profile [10] describing this predicted performance. The performance profile information is used to construct an abstract meta-level task structure in the form of a hierarchical task network. In our examples, we represent this as a TÆMS model [11]. We discuss some of the issues involved with each of these steps below.

Performance Profiles The content of a performance profile may vary according to the deliberative action associated with it. For instance, a performance profile associated with running a scheduler with certain parameters may contain an (expected schedule quality, expected scheduler runtime) pair. However, in the case of coordination algorithms, one may want to predict other features such as the expected number of messages passed. In general, though, one may define abstract quality, cost, and duration functions for each deliberative action which take the relevant factors into account.

Obtaining the data for the performance profiles raises another issue for the system designer. In some cases, performance profiles may be assumed a priori (as in [3]), supposing the designer has a deep enough understanding of the problem domain and the characteristics of the deliberative algorithms available to the agent. In other cases, the performance profiles must be obtained empirically, by letting the agent solve many sample problems using a variety of settings for the deliberative actions. The system designer must decide how much of the problem space to sample and how many experimental runs to perform. For many domains, this can be very time-consuming. One idea to handle this problem is to generate a smaller initial sampling of tasks to use in building the performance profiles (offline learning) and later update the profiles as the agent encounters more tasks (online learning). Of course the downside to this approach is that the agent's initial meta-control may be very ineffective.

Problem Classification Once the performance profiles have been designed, the system designer must define several "performance types" or categories into which to classify problems. For example, the designer may want to define a group for problems with an expected utility of 0-9, 10-19, etc. Alternatively, the designer may be able to use some clustering algorithms [12] on the performance profile data to automatically generate performance types. Once the performance types have been defined, each of the training samples used in generating the performance profiles can be labeled with the appropriate type and used to train a machine learning algorithm to predict the performance type of novel problems (for a discussion of various machine learning algorithms, see [13]).

The system designer must also define which problem features are relevant for classification. These features should be readily available to the agent or else easy to compute. For example, in our experiments where the agent's problems are specified in the TÆMS language, we extract certain high-level problem features from the task structure itself, such as how many tasks the agent can perform and the amount of temporal overlap between these tasks. Finally, the designer must make several decisions about the classification algorithm itself. For example, if the goal were to derive human-understandable meta-control heuristics for manual tweaking of the agent's deliberative algorithms, then a decision tree would be useful. Another issue is the robustness of the classifier to overfitting, especially important if there are only a small number of training instances. The designer may also want to use a classifier which allows incremental updates rather than batch training so that the agent may update itself after each experimental run.

Building the MetaAlternatives Task Structure. This section describes how we build the abstract meta-level TÆMS task structure, called "MetaAlternatives," once we have classified the incoming problems. We assume in the following example that the meta-level controller is trying to decide among several possible scheduler settings (modes) available to the agent.

In the TÆMS language, quality propagates upward through the task network based on quality accumulation functions (QAFs). For example, a Max QAF means that a task will achieve the maximum of the qualities achieved by its subtasks, and a Sum QAF means that a task will achieve the sum of these qualities. Tasks at the lowest level of the network, known as methods, are characterized by discrete quality and duration distributions. Quality in this case is an abstract concept and may mean different things for different domains.

Suppose our agent is currently in a scenario S consisting of the problems P1 and P2. For each problem P_i, we construct a subtask ($Problem_i$) that uses a Max QAF because our agent will accumulate quality only from the schedule that it actually executes. The methods used to accomplish this task correspond to the scheduler modes, for example SchedulerA, SchedulerB, SchedulerC. The task structure for scenario S has a root task called MetaAlternatives with two subtasks, one for each problem, combined together by a Sum QAF. The resulting task structure is shown in Figure 3. The MetaAlternative task structure is sent to the Markov Decision Process (MDP) sub-component as described in the next section, and the meta-level control policy is then computed.

2.2 Meta-Cognition Decision Process.

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 [14]. 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 a MDP is defined via its state set S, action set A, transition probability matrices P, and reward

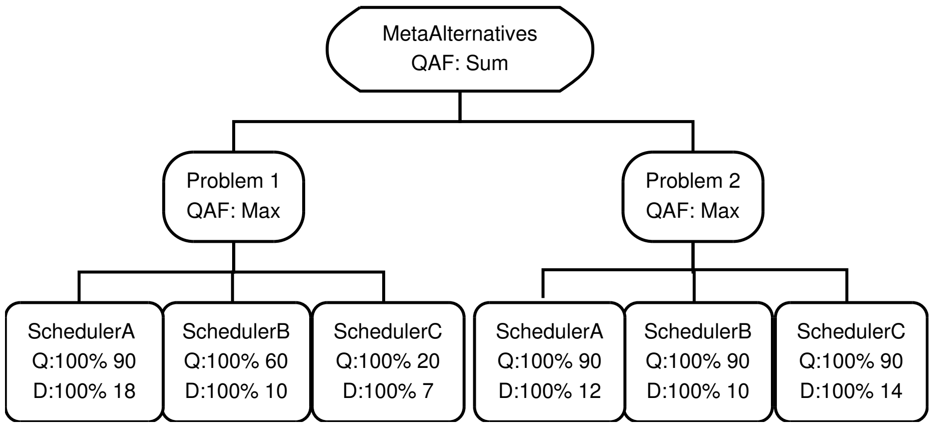

Fig. 3. A simple example of MetaAlternatives. Q and D represent distributions of quality and duration, respectively (for simplicity, in this example we assume that there is no uncertainty in quality and duration). Each $Problem_i$ task has the Max QAF (Quality Accumulation Function), indicating it gets the maximum quality of any of its subtasks. The top-level task has a Sum QAF, since its quality is the sum of quality for both problems.

matrices R. 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 π from the set of states S to the set of actions 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 [15]:

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

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

The process of generating an MDP from the MetaAlternatives task structure is based on the TÆMS to MDP translation algorithm in [16]. The resulting MDP is defined as follows: state in the MDP representation is a vector which represents the TÆMS methods that have been executed in order to reach that state along with their execution characteristics (quality and duration). The MDP action set is the set of TÆMS methods (executable leaf nodes). MDP actions have outcomes and each outcome is characterized by a 2-tuple consisting of discrete quality and duration values obtained from the expected performance distribution of the MDP action. The transition probabilities are obtained from the probability distributions of the corresponding MDP action as described in [16]. We assume that the agent has a limited amount of time to take deliberative actions, which determines the horizon of the MDP (a new MDP would be created during each decision-making epoch). The rewards are computed by applying a complex criteria evaluation function of the quality, cost and duration values obtained by the terminal states. The output from the MDP Solver will be an optimal policy that solves the MDP. Once the optimal policy is obtained, the meta-control component will determine the current state of the agent using the Current State Evaluator and the action corresponding to the current state is obtained from the optimal policy. When the action completes execution, meta-control will be notified; it will then recompute the current state and determine the current best action. This process continues until a terminal state is reached in the MDP or a new problem arrives that requires the meta-controller's attention.

3 Multiagent Extensions

The single-agent framework may be sufficient when agents are more or less independent; however, extending our framework to the more complicated case when cooperative agents must work together toward common goals creates its own challenges. In this section, we discuss the meta-control issues that need to be addressed in a multiagent context.

3.1 Problem-solving Contexts

First, there is the question of problem solving contexts. A problem solving context contains agent state information and other data required for decision making In our work, we have identified at least two types of contexts: current context and pending context. The agents' context when it is in the midst of execution is called the current context. A pending context is one where an agent deliberates about various what-if questions related to coordination with other agents. At any point in time, an agent has one current context and may have one or more pending contexts. When an agent is assigned a task, it creates a pending context where deliberative activities such as negotiation, unrolling the (MDP) search space, and policy computation are performed. The meta-control component will assist the agent in determining how much time and resources to allocate to each deliberative activity. This in turn will determine when an agent will terminate deliberation and begin execution by copying the commitments and execution policy produced by a pending context to the current context. Another meta-control issue when there are multiple pending contexts is to determine which pending context should be allocated resources for deliberation.

3.2 Increased Uncertainty from Other Agents

The second complicating factor is that the value of an agent's local decisions depends on the global state, thus the agent's decisions are affected by those of the other agents. This means that, unlike in the single agent case where the uncertainty in domain models is captured by static discrete distributions, the uncertainty over what other agents will do (at both domain and deliberation levels) is at least partially dependent on the ongoing decisions of the other agents. For example, an agent's meta-control needs to figure out how much time to spend deliberating over decisions that would follow from another agent changing the global state in way X, versus the time to spend thinking about what to do if the other agent does Y instead. The optimal deliberation policy should involve probabilities for X and Y, but these are dependent on the other agent. To cope with this uncertainty, agents need to reason about the actions of other agents, requiring the other agents to communicate some information about their respective policies. Thus an agent's optimal meta-level policy at some point might be to work on something else entirely until the other agents have done enough meta-level reasoning to get a good sense of what they will do regarding the various actions available. In fact, once an agent has communicated some information about its policy, it also needs to consider that some actions which might improve its local utility may actually have *negative* global utility, due to the cost of re-coordination with other agents at the meta-level and the need for other agents to re-deliberate. In essence, the agent must alternate between reasoning at the local level and at the global level. This necessitates the design requirement that the meta-control components should themselves be coordinated so that they direct local deliberation in ways that support collective deliberation.

3.3 Coordinated Meta-level Control

These agents may have multiple high-level goals from which to choose, but if two or more agents need to coordinate their actions, the agents' meta-control components must be on the same page. That is, the agents must be reasoning about the same problem and may perhaps need to be at the same stage of the problem-solving process (e.g., if one agent decides to devote little time to communication/negotiation before moving on to other deliberative decisions while another agent sets aside a large portion of deliberation time for negotiation, the latter agent would be wasting time trying to negotiate with an unwilling partner). Thus if an agent changes the problem solving context it is focusing on, it must have a way to notify other agents with whom it may have interactions. This suggests that the meta-control component of each agent should have a multiagent policy, where the progression of what deliberations agents do, and when, needs to be choreographed carefully, and include branches that account for what could happen as deliberation (and execution) plays out. Determining the multiagent policy is a complicated problem since these policies are not expected to be either reward-independent or transition-independent [17], implying that the multiagent policy is not simply the union of all of the single-agent meta-control policies,

We are taking the first steps to solve this problem by making some simplifying assumptions. Our current emphasis recognizes that, in the agent's world, deliberation alternates between local (individual) deliberation about the policy that will best achieve goals and commitments, and multiagent deliberation about commitments agents should make to each other. Given this, we currently "hardwire" the blueprint of the multiagent deliberation policy into the agents, requiring that agents collectively move between the local and global deliberation modes, where they move into the global deliberation mode when one or more of them receives a new or changed task, and they all move into the local deliberation mode when they agree that their negotiation over commitments has finished. In the future, we would want these modes to be interleaved more finely (so that local deliberation can better affect what is being negotiated, and vice versa), but things can quickly get complicated if agents are uncoordinated in this such that, for example, one is immersed in local thinking while another is hanging waiting for it to reply to a negotiation message.

3.4 Meta-Control Messages

This discussion of the complications arising from multiagent meta-control suggests the need for some kind of meta-level message passing. Here there are important tradeoffs between the amount of communication (both the size and number of messages) and the resulting overhead, and the usefulness of such communication. The system designer must choose what kind of information is contained in a meta-level message. Certainly, if agents can deliberate about several possible problem-solving contexts and must coordinate on these contexts, they should be able to communicate their choices to the other agents. However, the question remains as to what information about the agent's context should be communicated. In some situations, it may be enough for the agent to simply let the others know that it is thinking about context X; but in other cases, for instance when the agents are more tightly coupled, an agent may need to communicate some partial results of its current thinking as well.

Agents must also reason about how to handle meta-control messages from others. As mentioned earlier, we do not want a case where one agent is waiting on an acknowledgement of a message while the message recipient blissfully ignores the message and continues its own deliberations. On the other hand, it may not be desirable for one agent continually to interrupt the other agents' deliberations with its own meta-control messages. Continuing with the view of the meta-control layer as an abstraction of the deliberative layer, perhaps similar approaches can be successfully applied to meta-level message passing as to more traditional agent coordination problems.

4 Conclusion

As a conclusion, we briefly summarize some of the factors influencing the design of an agent's meta-control component:

Single Agent Case

- How much time is available for learning performance profiles?
- What features of the problem domain affect the agent's deliberative actions?
- Must the resulting meta-control policy be human-understandable?
- How much time is available for meta-control?
- Will myopic meta-control decisions suffice?

Multiple Agent Case (In addition to the factors above)

- How strongly coupled are the agents; that is, can they be treated as in the single agent case, or do they require extensive coordination and cooperation?
- What is the degree of centralization; do the agents regularly synchronize, or only rarely as-needed?
- How expensive/reliable is inter-agent communication?

In this paper we have identified design paradigms as well as open problems for sophisticated meta-control in multiagent systems. Automated abstractions, collecting useful performance profiles, handling complex levels of uncertainty and coordinated meta-control are meta-control issues we will be studying in greater detail in the context of the DARPA COORDINATORs project. We plan to build theoretical models that will address these design paradigms and study the trade-offs made in various approximations of these models.

Acknowledgments

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.

References

1. Goldman, R., Musliner, D., Krebsbach, K.: Managing online self-adaptation in real-time environments. In: Lecture Notes in Computer Science. Volume 2614. Springer-Verlag (2003) 6–23
2. Horvitz, E.: Rational metareasoning and compilation for optimizing decisions under bounded resources. In: Proceedings of Computational Intelligence, Milan, Italy (1989)

3. Raja, A., Lesser, V.: Meta-level Reasoning in Deliberative Agents. Proceedings of the International Conference on Intelligent Agent Technology (IAT 2004) (September 2004) 141–147
4. Russell, S., Wefald, E.: Principles of metareasoning. In: Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning. (1989) 400–411
5. Ghavamzadeh, M., Mahadevan, S.: Learning to communicate and act using hierarchical reinforcement learning. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2004)
6. Cox, M.T.: Metacognition in computation: A selected research review. Artif. Intell. **169**(2) (2005) 104–141
7. Schut, M., Wooldridge, M.: The control of reasoning in resource-bounded agents. The Knowledge Engineering Review **16**(3) (2001) 215–240
8. Russell, S.J., Subramanian, D., Parr, R.: Provably bounded optimal agents. In: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93). (1993) 338–345
9. Puterman, M.L.: Markov decision processes - discrete stochastic dynamic programming.Games as a Framework for Multi-Agent Reinforcement Learning. John Wiley and Sons, Inc., New York (1994)
10. Hansen, E.A., Zilberstein, S.: Monitoring and control of anytime algorithms: A dynamic programming approach. Artif. Intell. **126**(1-2) (2001) 139–157
11. Decker, K.S., Lesser, V.R.: Quantitative modeling of complex environments. International Journal of Intelligent Systems in Accounting, Finance, and Management **2**(4) (December 1993) 215–234
12. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys **31**(3) (1999) 264–323
13. Mitchell, T.M.: Machine Learning. WCB/McGraw-Hill (1997)
14. Sutton, R., Barto, A.: Reinforcement Learning. MIT Press (1998)
15. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-Dynamic Programming. Athena Scientific, Belmont, MA (1996)
16. Raja, A., Lesser, V., Wagner, T.: Toward Robust Agent Control in Open Environments. In: Proceedings of the Fourth International Conference on Autonomous Agents, Barcelona, Catalonia, Spain, ACM Press (July, 2000) 84–91
17. Becker, R., Zilberstein, S., Lesser, V.R., Goldman, C.V.: Solving transition independent decentralized markov decision processes. J. Artif. Intell. Res. (JAIR) **22** (2004) 423–455