Reasoning about Coordination Costs in Resource-Bounded Multi-Agent Systems

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

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

Abstract

Deliberative agents operating in open environments must make complex real-time control decisions on scheduling and coordination of domain activities. These decisions are made in the context of limited resources and uncertainty about the outcomes of activities. In this paper, we show that reasoning explicitly about the cost of control and domain actions leads to significant improvement in the performance of a multi-agent system. An empirical reinforcement learning algorithm which supports this reasoning process is presented.

Open environments are dynamic and uncertain. Deliberative agents operating in these environments must reason about their local problem solving actions, coordinate with other agents to complete tasks requiring joint effort, plan a course of action and carry it out. These deliberations may involve computation and delays waiting for arrival of appropriate information. They have to be done in the face of limited resources, uncertainty about action outcomes and in real-time. Furthermore, new tasks can be generated by existing or new agents at any time. These tasks have deadlines where completing the task after the deadline could lead to lower or no utility. This requires an agent to interleave deliberation with execution of its domain activities.

The agent has to choose which deliberative actions to perform when and whether to deliberate or to execute domain actions that are the result of previous deliberative actions. To do this optimally, 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 ability to sequence domain and control actions without consuming too many resources in the process is called the meta-level control (MLC) for a resource-bounded rational agent. Our approach is to equip an agent with meta-level reasoning with bounded computational overhead.

We consider three classes of deliberative actions: information gathering actions, coordination actions and planning/scheduling actions. These actions, also called control actions, are non-trivial requiring exponential work in the number of domain actions.

The first type of deliberative actions are information gathering actions which involve gathering information about the environment which includes the state of other agents. This information is used to determine the relevant control actions. These actions do not use local processor time but they delay the deliberation process.

The second type of deliberative action, coordination, is the process by which a group of agents achieve their tasks in a shared environment. In this work, coordination is the inter-agent negotiation process that establishes commitments on completion times of tasks or methods. Finally, the third type of deliberative actions involve planning and scheduling. Planning is the process in which the agent uses beliefs about actions and their consequences to search for solutions to one or more high-level tasks(goals) over the space of possible plans. It determines which domain actions should be taken to achieve the tasks. Scheduling is the process of deciding when and where each of these actions should be performed. In this work, planning is folded into the scheduling process.

The problem with most single and multi agent systems (Boutlier 1999; Musliner 1996; Raja, Lesser, & Wagner 2000; Kuwabara 1996; Zilberstein & Mouaddib 1999) is that they do not explicitly reason about the cost of deliberative computation. Hence, these systems have no way to trade-off the resources used for deliberative actions and domain actions. 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 (Simon 1976). We address this problem using a reinforcement learning-based approach.

There has been a variety of work on meta-level control (Simon 1982; Harada & Russell 1999; Stefik 1981) but in reviewing the literature there is little that is directly related the meta-level control problem discussed in this paper. The difficult characteristics of our problem are the complexity of the information that characterize system state; the variety of responses with differing costs and parameters available to the situation; the high degree of uncertainty caused by the non-deterministic arrival of tasks and outcomes of primitive domain actions; and finally the fact that the consequence of decisions are often not observable immediately and may have significant down-stream effects. The problem worked on that is closest to the complexity of our meta-level control decisions is the Guardian system(Hayes-Roth *et al.* 1994).

However their system is knowledge intensive and the heuristic rules seem very domain-dependent in comparison to the domain-independence of our approach. Although (Hansen & Zilberstein 1996) and (Russell & Wefald 1992) are applicable to this work, the techniques used are limited to specific problem solving situations that were much more structured than those encountered in our domain.

The intent of this research is to show that a meta-level reasoning component with bounded and small computation overhead can be constructed that significantly improves the performance of individual agents in a cooperative multi-agent system. This paper is structured as follows: We first describe an example scenario which motivates the meta-level questions addressed in this work, including the need to reason about control costs. This scenario exhibits partial observability, non-stationarity and action outcome uncertainty which are characteristic of multi-robotic systems. A formal description of the problem is then presented; followed by the empirical reinforcement algorithm which supports meta-level control reasoning. Finally experimental results showing the effectiveness of meta-level control are provided using a hand-generated heuristic approach as well as the reinforcement learning approach.

Motivating Example

In order to provide a clear picture of these five decisions described above, consider the simple scenario consisting of two rovers *Fred* and *Barney*. Rovers are unmanned vehicles equipped with cameras and a variety of scientific sensors for the purpose of planetary surface exploration. The discussion here will specifically focus on the various meta-level questions that will have to be addressed by *Fred*. Figure 1 describes *Analyze Rock*, also called task *T0*, and Figure 2 describes *Explore Terrain*, also called task *T1*, which are the tasks performed by *Fred*.

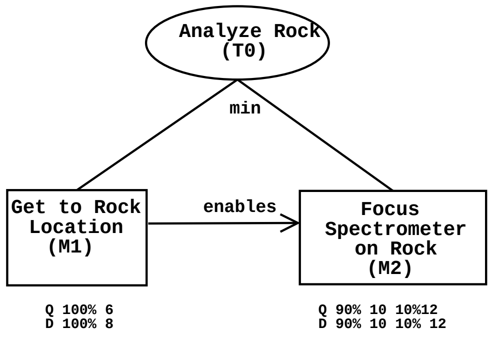

Figure 1: Fred's *Analyze Rock* task

In this example, each top-level task is decomposed into two executable primitive actions. In order to achieve the task *Analyze Rock* , *Fred* must execute both primitive actions *Get To Rock Location* and *Focus Spectrometer on Rock* in sequence. All primitive actions called *methods*, are statistically characterized in three dimensions: quality, cost and duration. Quality is a deliberately abstract domain-independent concept that describes the contribution of a particular action to overall problem solving. Thus, different applications have different notions of what corresponds to model quality. The sequence is denoted by the enables arrow between the two actions and the min quality attribution factor (which denotes a conjunction operator) states that the minimum of the qualities of the two actions will be attributed to the *Analyze Rock* task. To achieve the task *Explore terrain*, *Fred* can execute one or both primitive actions *Examine Terrain* and *Collect Samples* within the task deadline and the quality accrued for the task will be cumulative (denoted by the *sum* function).

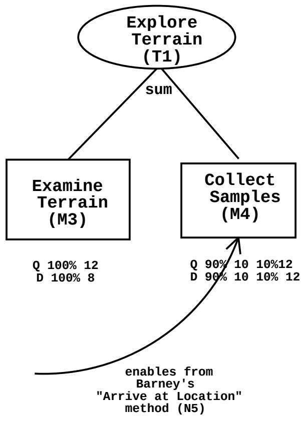

Figure 2: Fred's *Explore Terrain* task

Barney is equipped with a storage compartment while Fred is not. The *Collect Samples* method requires Fred and Barney to coordinate: Fred has the ability to pick up the soil sample and put it in Barney's storage compartment. This relationship between the two agents is denoted by the non-local *enables* from Barney's *Arrive at Location Method (N5)* method (*Barney's* task structure is not shown) to Fred's *Collect Samples* method. Utility and duration distributions for each primitive action is provided.

The following are sample meta-level questions that will be addressed by agent *Fred* and the cost-benefit analysis for only two of them are provided in the interest of space. The remaining questions are reasoned about in a similar fashion.

1. Should method *Collect Samples*, which is enabled by *Barney's* method *Arrive at Location*, be included in the *Fred's* schedule?
   Benefit: If method *Collect Samples* is included in *Fred's* schedule, *Fred* can increase its total utility.
   Cost: This implies that *Fred* and *Barney* have to negotiate over the completion time of *Barney's* method *Arrive at Location* and this will take time.
2. If *Fred* decides to negotiate, it should also decide whether to negotiate by means of a single step or a multi-step protocol (Lesser & Zhang 2002) that may require a number of negotiation cycles to find an acceptable solution or even a more expensive search for a near-optimal solution. For example, should a single shot protocol which is quick but

has a chance of failure be used or a more complex protocol which takes more time and has a higher chance of success.

Benefit: If *Fred* receives high utility as a result of completing negotiation on finish time of *Arrive at Location*, then better the protocol, the higher the probability that the negotiation will succeed.

Cost: The protocols which have 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).

3. If the negotiation between *Fred* and *Barney* using a particular negotiation protocol fails, should *Fred* retry the negotiation with *Barney* again?
4. Should *Fred* schedule newly arriving task *Tx* at arrival time or postpone scheduling to sometime in the future or drop that particular instance of the task.
5. When *Fred's* scheduler is called, it has to decide how much effort to invest in scheduling. Also how flexible should the schedule produced by the detailed scheduler be? How much slack should be inserted in the schedule?
6. When *Fred's* schedule deviates from expected performance by threshold α, should a reschedule be invoked automatically?

Formal Model

The meta-level controller (MLC) in making its decisions does not directly use the information contained in the agent's current state. This would include detailed information regarding the tasks that are not yet scheduled, status of tasks that are partially executed, and the schedule of primitive actions that are to be executed. Instead the MLC uses in it decision making, a set of high-level qualitative features that are computed from the full state information and pre-computed information about the behavior of the tasks that the system can handle. The advantage of this approach is that it simplifies the decision making process and provides the possibility for learning these rules (which we are currently exploring). The following are two examples of the high-level features. They take on qualitative values such as high, medium and low.

F1: Utility goodness of new task describes the utility of a newly arrived task based on whether the new task is very valuable, moderately valuable or not valuable in relation to other tasks being performed by the agent.

F2: Deadline tightness of a new task describes the tightness of the deadline of a new task in relation to expected deadlines of other tasks. It determines whether the new task's deadline is very close, moderately close or far in the future.

A detailed description of eleven features representing state is provided in (Raja 2003). The following is a decision theoretic formulation of the meta-level control problem.

1. Let *S* be the set of states of the agent and $s_i \epsilon\ S$ denote the agent state at stage i, $i = 0, 1, 2, 3 \ldots, n$
2. *A* is the set of possible control actions and $a_i \epsilon\ A$ is the action taken by the agent in state s_i.

   Control actions do not directly affect the utility achieved by the agent since they affect only the agent's internal state. These actions consume time and have only indirect effects on the external world.

   Control actions are followed by the execution of utility achieving domain actions. These domain actions are directly the result of control actions in the current and preceding states. These domain actions are not explicitly represented in this model since they are encased by the control actions.
3. A policy π is a description of the behavior of the system. A stationary meta-level control policy $\pi : S \rightarrow A$ specifies, for each state, a control action to be taken. The policy is defined for a specific environment.

   An environment is defined by three distributions describing task type, task arrival rate and task deadline tightness. Meta-level control is the decision process for choosing and sequencing control actions. In this work, there are five event triggers which invoke the meta-level control process. The occurrence of any of the triggers interrupts any other activity the agent in currently engaged in and control is shifted to the meta-level controller.
4. s_j is the new state reached when the agent is interrupted by an event requiring meta-level control reasoning while executing control action $\pi(s_i)$ followed by the execution of corresponding domain actions that follow $\pi(s_i)$.
5. $R(s_i, \pi(s_i), s_j)$ is the reward obtained in state s_j as a consequence taking control action $\pi(s_i)$ in state s_i and then executing the domain actions that follow $\pi(s_i)$.

   The reward is the cumulative value of the tasks and domain actions which are completed between the state transitions. Since the values achieved by the tasks have associated uncertainties, the reward function is represented as a distribution.
6. $U_\pi(s_i)$ is the utility of state s_i under policy π.
7. $P(s_j|s_i, \pi(s_i))$ is the probability that agent is in state s_j as a result of taking action $\pi(s_i)$ which is the action prescribed by policy π in state s_i.

The above model defines a finite Markov decision process (Bertsekas & Tsitsiklis 1996).

According to decision theory, an optimal action is one which maximizes the agent's expected utility, given by

$$E[U_\pi(s_i)] = E[\sum_{j=1}^{n} \gamma^j\ R(s_i, \pi(s_i), s_j)]$$

$\gamma \epsilon [0, 1)$ is a discount-rate parameter which determines the present value of future utility gains.

which can be computed as follows

$$E[U_\pi(s_i)] = \sum_{j=1}^{n} P(s_j|s_i, \pi(s_i))[R(s_i, \pi(s_i), s_j) + \gamma^j\ E[U_\pi(s_j)]]$$

The meta-level control problem is to find a best meta-level control policy π^* which maximizes the expected return for all states. This optimal policy can be found using dynamic programming (Bertsekas & Tsitsiklis 1996) and reinforcement learning (Sutton & Barto 1998) methods. These methods will implicitly determine the transition probability model and reward function defined previously.

In this work, the complexity of the state space makes it difficult to find the optimal policy. So an approximate meta-level control policy is found using a abstract state representation which will capture only the information relevant to the decision making process.

Reinforcement Learning

In the Reinforcement Learning (RL) framework, the learning agent interacts with an environment over a series of time steps t=0,1,2,3... At each time t, the agent observes the environment states, s_t, and chooses an action, a_t, which causes the environment to transition to state s_{t+1} and to emit a reward, r_{t+1}. In a Markovian system, the next state and reward depend only on the preceding state and action, but they may depend on these in a stochastic manner. The objective of the agent is to learn to maximize the expected value of reward received over time. It does this by learning a (possibly stochastic) mapping from states to actions called a *policy*. More precisely, the objective is to choose each action a_t so as to maximize the expected return, $E(\sum_{i=1}^{\infty} \gamma^i r_{t+i+1})$, where $\gamma \epsilon [0, 1)$ is a discount parameter. A common solution strategy is to approximate the optimal action-value function, or Q-function, which maps each state and action to the maximum expected return starting from the given state and action and thereafter always taking the best actions.

We adopt the learning approach developed in (Singh *et al.* 2000) for using RL in the design of a spoken dialogue system. Their problem is similar to ours in that it is also a sequential decision making problem and there is a bottle neck associated with collecting training data. Each of our simulation runs takes approximately four minutes since we are accounting for real-time control costs.

As described earlier, the MLC in making its decisions does not directly use the information contained in the agent's current state. The state of the markov-decision process is an abstraction of the actual state of the systems and uses the features described previously. We then specified the appropriate actions to take in each state. The actions are a list of control actions. The reward function is the sum of the utilities accrued by each completed task. The meta-level control policy is a mapping from each state to an action.

We then implemented an initial meta-level control policy which randomly chooses an action at each state and collects a set of episodes from a sample of the environment. Each episode is a sequence of alternating states, actions and rewards. As described in (Singh *et al.* 2000), we estimated transition probabilities of the form $P(s'|s, a)$, which denotes the probability of a transition to state s', given that the system was in state s and took action a from many such sequences. The transition probability estimate is the ratio of the number of times in all the episodes, that the system was in s and took a and arrived at s' to the number of times in all the episodes, that the system was in s and took a irrespective of the next state. The Markov decision process (MDP) model representing system behavior for a particular environment is obtained from state set, action set, transition probabilities and reward function. The efficiency of the model depends on the extent of exploration performed in the training data with respect to the chosen states and actions. In the final step we determine the optimal policy in the estimated MDP using the Q-value version of the standard value iteration algorithm (Sutton & Barto 1998). The expected cumulative reward (or Q-value) Q(s,a) of taking action a from state s is calculated in terms of the Q-values of successor states via the following recursive equation (Sutton & Barto 1998):

$$Q(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q(s', a')$$

The algorithm iteratively updates the estimate of $Q(s, a)$ based on the current Q-values of neighboring states and stops when the update yields a difference that is below a threshold. Once value iteration is completed, the optimal meta-level control policy (according to the estimated model) is obtained by selecting the action with the maximum Q-value at each state. To the extent that the estimated MDP is an accurate model of the particular environment, this optimized policy should maximize the reward obtained in future episodes. The summary of the proposed methodology is as follows

1. Choose an appropriate reward measure for episodes and an appropriate representation for episode states.
2. Build an initial state-based training system that creates an exploratory data set. Despite being exploratory, this system should provide the desired basic functionality.
3. Use these training episodes to build an empirical MDP model on the state space.
4. Compute the optimal meta-level control policy according to this MDP.
5. Reimplement the system using the learned meta-level control policy

Experiments

The agents in this domain are in a cooperative environment and have approximate models of the others agents in the multi-agent system. The agents are willing to reveal information to enable the multi-agent system to perform better as a whole. The interaction between 2 agents *Fred* and *Barney* is studied. The multi-agent aspect of the problem arises when there is task requiring coordination with another agent. The agent rewards in this domain are neither totally positively correlated(team problem) nor are they totally negatively correlated(zero-sum game). Multi-agent reinforcement learning has been recognized to be much more challenging than single-agent learning, since the number of parameters to be learned increases dramatically with the number of agents. In addition, since agents carry out actions in parallel, the environment is usually non-stationary and often non-Markovian as well (Mataric 1997). The experiments describe results on the convergence rates of the policies of the two agents in simple scenarios.

The meta-level control decisions that are considered in the multi-agent set up are: when to accept, delay or reject a new task, how much effort to put into scheduling when reasoning about a new task, whether to reschedule when actual execution performance deviates from expected performance, whether to negotiate with another agent about a non-local task and whether to renegotiate if a previous negotiation falls through. For all the experiments, the following costs are assumed. The meta-level control actions have an associated cost of 1 time unit; the drop task and delay task actions take 1 time unit also. The decision to negotiate and whether to renegotiate also take 1 unit of time. The call to simple scheduler costs 2 time units and the cost of computation of complex features costs 2 time units, the cost of detailed scheduling tasks with less than five methods is 4 units, with less than ten methods is 12 time units and greater than ten methods is 18 time units.

The task environment generator in the multi-agent setup also randomly creates task structures while varying three critical factors:

1. complexity of tasks $c \epsilon \{simple, complex, combo\}$
2. frequency of arrival $f \epsilon \{high, medium, low\}$
3. tightness of deadline $dl \epsilon \{tight, medium, loose\}$.

Complexity of tasks as described earlier refers to the expected utilities of tasks and the number of alternative plans available to complete the task. A simple task, in the multi-agent setup, has two primitive actions and its structure and number of possible alternatives is similar to the Analyze-Rock task (Figure 1). The utility distribution and duration distribution of a simple task is within a 5% range of the corresponding distributions of AnalyzeRock. A complex task has a structure similar to ExploreTerrain task described in Figure 1.

We first tested the effectiveness of meta-level control using two context sensitive heuristic strategies: the Naive Heuristic strategy (NHS) that uses myopic information to make meta-level control action choices; and the Sophisticated Heuristic strategy (SHS) that uses current state information and predictive information about the future to make non-myopic action choices. These were compared to base-line approaches which used a random and deterministic strategy respectively. A detailed description of these strategies is provided in (Raja 2003).

The behavior of two interacting agents is presented in Figure 3 and Table 1. Performance comparison of the various strategies in an environment, AMM, over a number of dimensions are provided. AMM is characterized by a combination of tasks (A), medium frequency of arrival (M) and medium deadline tightness(M). The results show that the combined utilities of the two agents when using the heuristic strategies is significantly higher than the combined utilities when using the deterministic and random strategies. The utility obtained from using SHS is significantly higher than NHS and also 14% more tasks are completed using SHS than the NHS. The improvement in utility can be explained by the drop in control time (Row 3) since the heuristic strategies choose control methods which are more appropriate for the context and free up resources for other domain activities. These preliminary results are encouraging. Further experimental studies to establish the advantage of meta-level control in multi-agent systems are ongoing.

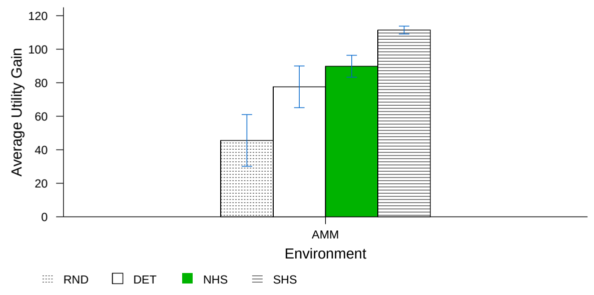

Figure 3: Average Utility Comparison between Heuristic Strategies and Baseline Strategies in a Multi-agent environment. The error bars are one standard deviation above and below each mean

Row#		SHS	NHS	Deter.	Rand.
1	AUG	111.44	89.84	77.56	45.56
2	σ	2.33	6.54	12.45	15.43
3	CT	9.21%	8.09%	14.28%	7.15%
4	RES	0%	14.28%	19.93%	1.49%
5	PTC	71.32%	56.34%	54.17%	57.78%
6	PTDEL	8.8%	3.98%	0%	59.96%

Table 1: Performance evaluation of four algorithms for two agents in a environment AMM with a combination of tasks, medium frequency of arrival and medium deadline tightness. Column 1 is row number; Column 2 describes the various comparison criteria; Columns 3-6 represent each of the four algorithms; Rows 1 and 2 show the average utility gain (AUG) and respective standard deviations (σ) per run; row 3 shows the percentage of the total 500 units spent on control actions(CT); row 4 is percent of tasks rescheduled (RES); Row 5 is the percent of total tasks completed (PTC);Row 6 is percent of tasks delayed on arrival (PTDEL)

Preliminary experimental results describing the behavior of the two interacting agents is presented in Figure 4 and Table 2. Agent *Fred's* was fixed to the best policy it was able to learn in a single agent environment. Agent *Barney* then learned its meta-level control policy within these conditions.

Performance comparison of the heuristic strategies to the RL strategy in environment AMM is provided. The results show that the combined utilities of the two agents when using the RL strategy is as good as the SHS strategy which uses environment characteristic information in its decision making process. The RL strategy also learns policies which significantly outperform the NHS strategy in this environment. Further experimental studies to establish the advantage of automatically learning meta-level control policies in multi-agent systems are ongoing.

The experimental evaluation lead to the following con-

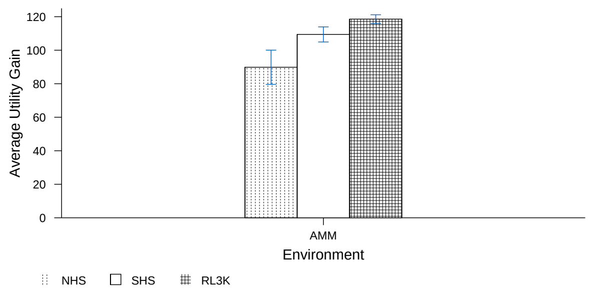

Figure 4: Average Utility Comparison between Heuristic Strategies and RL Strategy (300 training episodes) in a Multiagent environment. The error bars are one standard deviation above and below each mean

Environment	RL-3000	SHS	NHS
AMM-UTIL	118.56	111.44	89.84
AMM-CT	8.86%	9.21%	8.09%

Table 2: Utility and Control Time Comparisons over four environments; Column 1 is the environment type; Column 2 represents the performance characteristics of the RL policy after 3000 training episodes; Column 3 and 4 represent the performance characteristics of SHS and NHS respectively;

clusions : Meta-level control reasoning is advantageous in resource-bounded agents in environments that exhibit non-stationarity, action outcome uncertainty and partial-observability; the high-level features described in the previous chapter are good indicators of the agent state and facilitate effective meta-level control; the heuristic strategies are good indicators of the positive effects of meta-level control in resource-bounded agents because they outperform deterministic and random strategies; and predictive information about future arrival tasks is useful in some environments and not in others.

We describe a reinforcement learning approach which equips agents to automatically learn meta-level control policies. The empirical reinforcement learning algorithm used is a modified version of the algorithm developed by (Singh *et al.* 2000) for a spoken dialog system. Both problem domains have the bottle neck of collecting training data. The algorithm optimizes the meta-level control policy based on limited training data. The utility of this approach is demonstrated experimentally by showing that the meta-level control policies that are automatically learned by the agent perform as well as the carefully hand-generated heuristic policies.

We are currently studying the effectiveness of this approach in environments with 10's of agents. We are also using insight gathered from the heuristic approaches and reinforcement learning approaches to study the effectiveness parameters which will allow for policy generalization over similar environments. And finally, we plan to reason about organizational adaptation and communication as control actions to achieve our overall goal of accurately reasoning about costs at all levels in large-scale, cooperative multi-agent systems.

References

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

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

Hansen, E. A., and Zilberstein, S. 1996. Monitoring anytime algorithms. *SIGART Bulletin* 7(2):28–33.

Harada, D., and Russell, S. 1999. Extended abstract: Learning search strategies. In *Proc. AAAI Spring Symposium on Search Techniques for Problem Solving under Uncertainty and Incomplete Information, Stanford, CA, 1999.*

Hayes-Roth, B.; Uckun, S.; Larsson, J.; Gaba, D.; Barr, J.; and Chien, J. 1994. Guardian: A prototype intelligent agent for intensive-care monitoring. In *Proceedings of the National Conference on Artificial Intelligence*, 1503–1511.

Kuwabara, K. 1996. Meta-level Control of Coordination Protocols. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS96)*, 104–111.

Lesser, V., and Zhang, X. 2002. Multi-linked negotiation in multi-agent system. *Proceedings of the First International Joint Conference on Auton omous Agents And MultiAgent Systems (AAMAS 2002)* 1207–1214.

Mataric, M. 1997. Reinforcement learning in the multi-robot domain.

Musliner, D. J. 1996. Plan Execution in Mission-Critical Domains. In *Working Notes of the AAAI Fall Symposium on Plan Execution - Problems and Issues*.

Raja, A.; Lesser, V.; and Wagner, T. 2000. Toward Robust Agent Control in Open Environments. In *Proceedings of the Fourth International Conference on Autonomous Agents*, 84–91. Barcelona, Catalonia, Spain: ACM Press.

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

Russell, S., and Wefald, E. 1992. *Do the right thing: studies in limited rationality*. MIT press.

Simon, H. 1976. From substantive to procedural rationality. 129–148.

Simon, H. A. 1982. *Models of Bounded Rationality, Volume 1*. Cambridge, Massachusetts: The MIT Press.

Singh, S. P.; Kearns, M. J.; Litman, D. J.; and Walker, M. A. 2000. Empirical evaluation of a reinforcement learning spoken dialogue system. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 645–651.

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

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

Zilberstein, S., and Mouaddib, A.-I. 1999. Reactive control of dynamic progressive processing. In *IJCAI*, 1268–1273.