# An MDP Approach for Agent Self Monitoring

**Ping Xuan**
Department of Mathematics and Computer Science
Clark University
950 Main Street
Worcester, MA 01610
pxuan@clarku.edu

### Abstract

An emerging trend in multiagent system research is the use of decentralized control models (notably the decentralized MDP formalism) as the basis for decision theoretic studies of multiagent problem solving. Research often focuses on the development of solution algorithms for some special classes of decentralized MDPs. While these studies will no doubt enhance our understanding of the formal foundations for multiagent system, they have a long way to go toward the actual making and deployment of decision-theoretic agents. This leads to the need for some new thinkings toward the role of MDPs in multiagent research. Since MDP policies define the mapping between an agent's knowledge (information states) and its actions, they can be viewed as abstractions to, or be regarded as an external view of, the internal agent reasoning processes. As such, we propose the use of MDPs as a tool for agents to establish meta-level capabilities such as self-monitoring and self-control. These capabilities are essential for agents to learn organizational knowledge. Furthermore, we also propose to apply this MDP formalism to the group level, to represent the group control and coordination knowledge.

## Introduction

Decision theoretic studies of multiagent problem solving have recently become one of the most active fields in multiagent systems research. Most models are based on some types of the Markov Decision Processes (MDPs) (Puterman 1994). The centralized multiagent system model (which is described in terms of global states and joint actions) proposed by Boutilier (Boutilier 1999) uses standard MDPs, and the decentralized models (Xuan, Lesser, & Zilberstein 2001; Bernstein *et al.* 2002; Pynadath & Tambe 2002) (which are based on each agent's partial observation of the global state and individual agent local actions) involve the creation of a new type of MDPs called the decentralized MDPs (DEC-MDPs). DEC-MDPs capture the decentralized nature of multiagent systems and try to solve the problem of choosing optimal local actions (including communication or information sharing actions) based on partial knowledge of the overall system in each agent. The solution of a DEC-MDP consists of one local policy (i.e. a mapping from the agent's local information states to the agent's actions) for each agent. Given the local policy, each agent's problem solving can be characterized as the following: the agent determines its local information state, then decides its action according to the policy, performs the action, observes the action outcome, and updates its information state and repeats this process until the problem solving finishes. The goal of solving a DEC-MDP is to obtain the maximum expected utility for the agent team involved in the problem solving (i.e. global rewards).

Although DEC-MDPs are successful in defining the decision theoretic foundation for multiagent problem and giving the definition of optimality, they are indeed very complex models and very hard to solve. The complexity class for DEC-MDPs is NEXP - usually means that double exponential time (e.g. $2^{2^n}$) would be required to obtain optimal solutions. Much of the recent research results in this field focus on obtaining optimal or approximate solutions for special classes of DEC-MDPs (Becker *et al.* 2003; Goldman & Zilberstein 2003; Bernstein & Zilberstein 2003; Nair *et al.* 2003; Chades, Scherrer, & Charpillet 2002). While these works are exciting in helping us taming the complexity issue, there is a long way to go toward the actual making and using of agents that are based on decision-theoretic reasoning processes. We believe that DEC-MDPs should play an important role in the design of multiagent systems, but it is only one link in the whole picture, not an end-to-end solution. The following describes some of the difficulties ahead in applying DEC-MDPs in agent organizations:

- Openness: first of all, like all decision theoretic planning frameworks, DEC-MDPs solve closed (e.g. well defined, completely quantified) problems. Open-ended processes that deal with unbounded amount of uncertainty have difficulty fitting into the DEC-MDP framework. Open systems often involve dynamic agent population, no *a priori* knowledge of environment characteristics, incomplete view of the world states, etc. Thus, it is unlikely to envision that decision theoretic techniques alone would address all these issues.
- Organizational flexibility: thus far, DEC-MDPs have not addressed issues such as self-organization and reorganization. While organizational structures may be manifested

in DEC-MDP policies (for example, some agents are allocated more tasks and more tightly coupled than some others), there is no group awareness or group representation.

- Stability: like typical MDPs, the optimal solution for DEC-MDPs is very sensitive to slight changes in the environment. This may present problem for organizational stability. So far, DEC-MDPs are used to to model one episode of problem solving, but agents organizations are meant for long term collaborations. In these cases, there may not be clear notion of the start or finish of problem solving.
- Structured problem solving: MDP policies are very low level representations of plans. The plan is exhaustive in detail and has only one flat level. The lack of structure for clearly reflecting task flows and task hierarchies makes the representation rather unappealing and also very hard to scale.
- Complexity and overheads: although in theory an agent only needs its local policy to carry out the problem solving (in a rather mechanical way), there is the question of how to obtain such a policy. Requiring each agent to independently compute the identical policy is obviously impractical and wasteful. Offline or centralized computation may be a possibility, but that would mean that the agents receiving their local policies really have very little control (or intelligence). In DEC-MDPs, the local policy alone does not enable one agent to reason about its role in the team work, since the reward structure is defined over global states and therefore to calculate the reward we need to know the local policies in all agents, not just one.

These difficulties lead us to explore the means of integrating decision theoretic approaches in agent organizations. We notice that one interesting difference between the MDP approach and other traditional multiagent planning frameworks is that the MDP policies do not talk about the agent's goals, intentions, desires, etc – these are really high meta-level constructs that reflect the agent's internal reasoning process. In fact, an MDP policy need not to have the notion of agent internal reasoning process at all - all is involved is one lookup. There is no direct or obvious line of reasoning to justify or interpret the agent action other than the simple belief that the policy is working toward a maximum (or close to optimal) expected global reward – for MDPs, those interpretations or internal reasoning processes are not important any more because the policy already completely defines the agent's behavior. In other words, the policy *is* the agent.

Generally, if an agent's policy is known to others, then the agent's behavior is predictable given the information state of the agent. As such, other agents will not be interested in the internal reasoning process employed in this agent. Conversely, if the agent can reveal information about the policy it uses, it does not need to reveal details of its internal reasoning process. This is very important in open multiagent systems because a truly open system must allow cooperation among heterogeneous agents. Here, the heterogeneity not only refers to the difference in the agents' capabilities, resources, and roles, but also is manifested in the different internal reasoning techniques and processes that agents use. Because MDPs are neutral to the different types of reasoning models, they can be used as a common representation for agents to reason about other agents' behaviors without the knowledge about other agents' internal reason process. However, this does not mean that the agents should post their MDPs and policies to each other (hence converting to decision-theoretic agents in effect) – that would be rather cumbersome as discussed above. Instead, we want to use MDPs to provide a level of abstraction for the agent's internal reasoning process, try to preserve the meta-level representation of the agent plans. If we can use MDPs to provide an encapsulated view of the agent plan, then this view can serve not only as a yardstick for self-monitoring, but also as the interface for other agents to understand its behavior. The key here is to use MDPs to represent the plan dynamics without tying to the process of making plans or the detailed plan (policy) itself. The evolution of a plan can be modeled in a stochastic process, and the meta level decisions made toward the plan can be regarded as a decision process. An MDP model that provides these information would be the first step toward the establishment of self-awareness in agent problem solving.

In this paper, we first provide a mechanism that uses MDP formalisms to encapsulate agent behavior, show how we can use it for agent self-monitoring and self control, and then extend it to the group level to represent coordination knowledge and provide a tool for group level monitoring.

## Agent Policy Abstractions

Our first step toward establishing a formalism for the agent plan dynamics is to establish abstract views of agent plans. This does not mean that we want to translate the agent plan into an MDP policy, although this is possible in most cases. Instead, we want to focus on meta-level views of the agent plan. In the following we first discuss the general approach, and then discuss a detailed example in the context of systems using traditional multiagent planning techniques.

Typically, a plan tries to achieve a certain goal. In the most trivial and abstract way, we can mark the beginning of the plan as a beginning state, where the plan is ready to be performed, and mark the finish point(s) of the plan as the finishing state(s), where the goal is achieved or failed (a plan can halt in the middle or branch out as a result of the stochastic nature of action outcomes.) The plan execution itself can be viewed as a transition between the two ends. The reward value can be assigned at the finish state(s). This would define a simple MDP for the plan. Alternative plans can be similarly represented via different transitions and different finishing rewards. Such a MDP models the process of achieving the intended goal and thus enables the agent to assess the problem solving situation online.

An agent may have multiple goals to achieve. In this case we may define separate MDPs for separate goals, and add a reward function to calculate the overall reward. Also, a goal may be decomposed int subgoals - this can also be modeled via MDPs for each subgoal, plus a function that relates the subgoal states reward to the goal state reward.

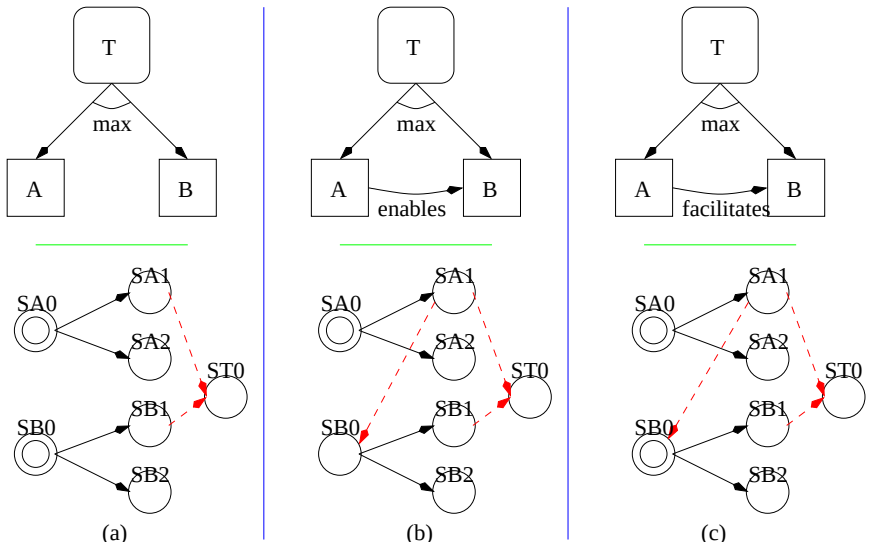


Figure 1: Model processes for simple task structures:
(a) simple subtasks *OR* relationship,
(b) with an *enables* interrelationship,
and (c) with a *facilitates* interrelationship

If we decide to add more the details, we can introduce more intermediate states to represent the partially completed status of the plan, e.g. states like "the first action in the plan succeeded", "the second action failed" etc. The transitions would then correspond to segments of the plan. The finishing reward can become of a function of the rewards of the intermediate states. We can continue this process until the transitions become a single atomic action.

The set of MDPs we obtained is what we call the meta level plan abstractions. They reflect the knowledge of the agent regarding how to achieve a certain goal (the plans), how the problem solving process may progress (the transitions and intermediate states), and how to evaluate the plan execution (the reward structure).

As an example, let us study Figure 1, in which we illustrate the MDPs for simple planning problems. The problems are illustrated using TAEMS task structures (Decker & Lesser 1993) - a type of hierarchical task networks often used in agent planning and coordination. In part (a) of Figure 1, the task structure in the top part consists of one high level task T, which can be decomposed into lower level subtasks A and B, and the *max* symbol indicates how subtask qualities (rewards) relate to the quality of high level tasks – in this case, that equals to the larger reward between the two subtask rewards. The lower part shows the MDPs (2 MDPs, in fact) that we obtained for representing meta level plan dynamics. The basic elements are the following:

- The first MDP consists of states SA0, SA1, and SA2. SA0 indicates the ready state for performing the task A (a beginning point), while SA1 and SA2 represents two outcomes of task A: success (SA1) and failure (SA2). Similarly, in the second MDP consisting of SB0, SB1, SB2, the same notations are used except that they involve task B instead of A. The state ST0 is the state to indicate that T has been accomplished (a finishing point).
- SA0 and SB0 are marked as double circle because they are both *ready* states. A ready state means that the agent can start executing actions in the MDP to pursue the goal *at any time*. It is important to note that an agent may start with one goal and later switch to a different goal, therefore first pursue the transition associated with SA0 (i.e. do task A), and then at a later time initiate the transition in SB0. Thus, multiple ready states implies the agent's option of pursuing multiple, concurrent goals.
- State SA1 and SB1 correspond to successful outcomes of task A and B. According to the task structure, both states thus also indicates that task T has been accomplished, which is essentially the meaning of the state ST0. In the figure, we use dashed lines to connect SA1 and SB1 with ST0, to indicate that both SA1 and SB1 are *aliases* of ST0. The reward of ST0 is the larger of the rewards in SA1 and SB1.
- The MDPs thus encodes the following information: to reach the goal ST0, we would need to reach either one or both aliases of ST0: SA1 and SB1 (we can denote these alias states as ST0:SA1 and ST0:SB1). Then we have one MDP each for ST0:SA1 and ST0:SB1, and each of them encodes a plan or a process for achieve their respective goals. There are also failure states SA2 and SB2 (with certain likelihoods), that do not lead to the completion of the goal T.

Part (b) of Figure 1 differs from part (a) only in one place: there is an *enables* relationship from task A to task B. In TAEMS, this relationship means that task A must be accomplished (succeed) before task B can start (essentially, a

precedence constraint). In this case, we note some changes in the resulting MDPs: first, SB0 is no longer a ready state, because it has a constraint need to be satisfied. Second, SA1 now becomes an alias state for SB0, because when SA1 is reached, task B can start, which means that the state SB0 becomes ready at that time. This change to the MDP incorporates the constraint quite easily.

Another type of interrelationships, *facilitates* (as illustrated in part (c) is a soft interrelationship – it differs from *enables* in that it does not prohibit task B from starting when task A has not finished, but if task A succeeds, its outcome is going to have a positive impact on task B (e.g. a higher resulting reward). In this case the state SB0 is again a ready state, however the alias SB0:SA1 still exist and the reward of SB1 would differ depending on whether the starting state is SB0 or SA1.

Note that the MDPs we obtained in this example are not necessary the MDPs needed for computing the decision theoretic policy. Rather, what we obtained really focus on the stochastic processes that describe the evolution of the problem solving goals. The different MDPs may have common states (aliases) that signifies the (sub)goal interactions. When the internal reasoning process decides which goals to pursue and which ones to ignore, our MDPs would reflect and track that decision.

## Agent Self Monitoring and Learning

In this section we discuss how we can use the MDPs obtained in the previous section in agent self monitoring and self control. Our intention is to use those MDPs together with the agent's actual plan (which is the product of the agent's internal reasoning process) to monitor the agent's problem solving process, i.e. a type of reflective execution monitoring. Such monitoring can also be performed by other agents in the agent organization to obtain interpretations about this agent's behavior and make predictions.

The role of the monitoring process is to examine the plan execution, assess the alignment between the current plan and the current situation, and to determine if the plan behaves as expected and if new plan needs to be formulated. Examples of conditions that warrant replanning include the following:

- Tasks do not behave as expected. For example, a task may take too long to finish, and the current plan does not take into account of such variation. Other examples include unexpected results, unresolved constraints, insufficient information, etc. These conditions indicate deficiency in our models, and they are easy to detect in the MDP framework, since they correspond to previously not known transitions, wrong state/transition properties, etc, in the MDP representation.
- Changes in the agent goals. New tasks may arrive, and some may be critical tasks. Also, the value and importance of current tasks may change in a dynamic environment. The current running plan may be outdated by these events. These can be modeled in our monitoring framework by the introduction of new MDPs (i.e. new goals) and modifications of existing MDPs.

For the purpose of this discussion, let us conceptualize the underlying agent planner as a dynamic, goal-driven planner that is capable of computing a new plan given a set of goals and their criteria. One instance of such planner is the Design-to-criteria (DTC) planner (Wagner, Garvey, & Lesser 1998) that is known for scheduling complex TAEMS task structures. For simplicity, let us also, assume that the new plans are generated instantaneously. In reality, planning process could be quite time consuming and therefore interfere with the execution process. This is known as an example of bounded rationality and beyond the scope of this work for now.

The basic monitoring process involves checking the MDPs against the current plan execution. Those MDPs define the model of execution, the expected behaviors and outcomes, the the evaluation criteria (the reward structure). They can be viewed as processes that are parallel to the execution process and are updated according to the outcomes of the execution process. As the problem solving progresses, the trajectories of agent activities are displayed in these MDPs. Specifically, the monitoring process may gather information through this representation to reflect upon its own behavior and support the following functions:

- **Status/progress reports.** The agent's trajectory in each MDP encodes information regarding the status of the plan during the pursue of the goal state. At any moment, when the agent is pursuing a goal, its status can be represented in the corresponding MDP as in one of the following three cases:

  1. The agent is in a state that is currently *ready*. This means that the agent may start to execute the actions and make the transition toward the next state, but has not started the action. This reflect the agent's decision to delay (idle) the actions toward the goal and temporarily shifting the focus to other goals.
  2. The agent is in a state that is currently not ready. This means that the agent is not ready to execute further actions toward the next state; it has to wait until the constraints are satisfied and the state becomes ready.
  3. The agent is in the middle of a transition. The actions associated with the transition are being performed and their status/progresses are periodically reported to the MDP.

  Signals can be raised when anomalies are detected, such as trying to perform actions that are not allowed/specified in the MDP, actions take longer to complete, unexpected outcomes, etc.

- **Focus tracking.** By studying which MDPs are in ready states, we can learn which goals the agent is currently focusing on and therefore learn if its plan prioritizes the goals correctly. Idle time may be strategic - for example, the agent may be anticipating more information to arrive shortly which may facilitates its next actions.
- **Prediction.** The MDP models allow the agent to predict possible future outcomes, including the estimated rewards, completion time, which can be used to provide

forecasts to other agents and plan ahead for future contingencies.

- **Bottleneck detection.** Identifying bottlenecks, overly constrained plans, or critical regions, is crucial in plan optimization. By monitoring the waiting time of MDP states, we can track parts of the MDPs that incur a long waiting time for other MDPs.
- **Model verification.** Sometimes, the agent's problem solving knowledge may not be complete accurate. This leads to inaccurate MDPs models and thus mismatch between the actual action outcome and expected behavior. Thus, we can use the monitoring process to correct our assumptions about the environment and verify the models.
- **Statistical (long term) learning.** For stochastic processes, the monitoring process should also verify the stochastic models used in the MDPs. For example, the distribution of task durations need to be monitored over repeated runs to verify if the model is correct. Furthermore, through statistical learning we may be able to establish models for previously unknown environment parameters. For example, an agent may want monitor the arrival rate of new tasks and use that knowledge in its planning.
- **Profiling and preference/pattern detection.** An important characteristic of intelligent beings is the ability to learn from past experiences and extract patterns. Also, agent needs to understand its own limitations and be able to provide quick reasonable assessments without resorting to full scale reasoning. Agents also need to be aware of its own bias in its reasoning process, since different types of planners are suitable for different types of problems. By monitoring the agent activities over a long periodic of time, the agent may be able to discover that certain types of problems are easier than others, or some goals/alternatives are rarely pursued.

As an example, let us consider an agent facing recurring requests for performing the task T illustrated in Figure 1 part (a). Suppose task A takes shorter average duration to run than task B, but its expected reward is also lower than task B. Also suppose that the reward for completing T diminishes after some deadline. In this case, it is trivial to construct an optimal plan for a single occurrence of task T, but such a plan may not be suitable if the requests arrive more frequently than the agent can handle. For example, it may be the case that 50% of the requests would expire (not processed by the agent) if the agent always try the optimal policy for the requests. In this case, when a plan expires, the monitoring process notices that the related MDPs are in ready state for too long and the rewards have already diminished. Those goals (to service those requests) would thus be abandoned. Replanning to address the overload issue would be required. Suppose the planner now comes up with a non-optimal plan, say only performing task A for the requests, and as a result 80% of the requests may be serviced. The agent now can list itself as capable of handling 80% of the load.

## Representation Group Coordination Activities

Agent self-monitoring and introspection is really the first step toward organizational learning. The next step is to represent coordination knowledge. That include information such as who is available, who has the capability, how accountable the other agents are, and what are each other's role during the collaboration. Again, we argue that it is not feasible to direct model other agent's internal reasoning, rather, we should rely on the external observations of the agents. The MDP models of the agent plan dynamics can serve this purpose. However, it should be noted that those plan dynamics models are private, and an agent does not need to post all those models to the agent organization, nor would it need to post the exact models – simplified models and/or relaxed models can be used, as long as they do not conflict with the agent's own models.

The semantics of posting such a model is closely related to the semantics of agent commitments. Typically, a commitment means a pledge to take certain specified course of action, e.g. "I promise to accomplish task T for you by time 20" (Jennings 1996). In this case, both agents would agree on the actions being promised and therefore a mutual consensus is established. It is based on commitments that joint goals and joint plans can be formed. It should be noted that commitments are dynamic objects and therefore have their own states and dynamics (Xuan & Lesser 1999), i.e., commitments may be modified, changed, or decommitted.

The posting of a model is intended to announce that *if* the goal indicated in the model is required in a commitment, then the model would describe the dynamics of that commitment. For instance, if the MDP involving states SA0, SA1, and SA2 in Figure 1 part (a) is posted, then this basically means that if this agent offers commitment on task A (or T), this MDP would represent the possible outcomes: it may fail (in state SA2) or succeed (SA1) and the reward can be calculated accordingly. This is essentially a statement about the agent's capability, since at this point no commitment has been established.

Those posted models address the issue of finding the capabilities of the other agents, but has not addressed the issue of representing and establishing commitments. Fortunately, there is a large body of research in coordination mechanisms that focus on this subject and can be easily adapted using our MDP formalism. Specifically, under the GPGP (Decker & Lesser 1992) framework, a family of coordination mechanism are developed to establish commitments in response to nonlocal task interrelationships. These interrelationships are just like the *enables* and *facilitates* interrelationships illustrated in part (b) and (c) of Figure 1, except that the tasks involved in the interrelationships belong to different agents. In this case, a commitment from the agent pledging task A would be proposed.

Again, we can use MDP formalisms to represent the commitments. The key idea is to note that commitments are quite similar to agent goals - they are, after all, joint goals. Just as there are the plans for achieving a goal, there are plan for honoring a commitment. While the commitment is being honored, the agent's posted capability is all that is needed to describe the evolution of the commitment: we can monitor

the status of the commitment the same way we monitoring the status of the goals. However, since commitments are social objects while goals are local, we need to explicitly model the possibility of decommitment, so that the agents can plan for contingencies decommitment happens. Thus, there will be a transition from the state "commitment being honored" to the state "commitment decommitted". The "commitment being honored" state essentially encapsulates the complete MDP for achieving the pledged task.

Decommitment may be an explicit action if the underlying plan specifies it. Otherwise, decommitment may be proved or predicted when the pledged actions do not progress as anticipated and thus render the fulfillment impossible. Note that the reasons for decommitment need not be specified in the model of commitments – they are information local to the decommitting agent and need not to be publicized.

Just like the MDP models for agent plan dynamics, the purpose of the MDPs for the commitments is to specify the expected behaviors of the agents involved, so that the agents can use them as a reference to monitor their compliance and provide the interpretation for their coordination activities. The same type of monitoring and learning ability can thus be extended to coordination activities.

## Conclusion and Future Directions

Organizational learning is a vast topic and this paper only address a small step in this line of research, namely the self-monitoring and coordination. This model is still in an infancy state and needs much work. One of the most difficult issues is to establish a three level architecture consists of individual level, group level, and the organization level. Such an architecture is fundamental for the study of organizational learning processes (Kirn 1996). So far our model is capable of dealing at individual level and at a fixed team level, but we have yet to address the issue of formulating group identities. In the future we will continue to explore the issue of long-term interactions and group formulation. However, we believe that the developing the capability of self monitoring alone is an important step and may find many applications, such as plan diagnosis and optimization.

## References


Becker, R.; Zilberstein, S.; Lesser, V.; and Goldman, C. V. 2003. Transition-independent decentralized markov decision processes. In *Proceedings of the Second International Conference on Autonomous Agents and Multi-agent Systems (AAMAS-03)*.

Bernstein, D., and Zilberstein, S. 2003. Generalized dynamic programming for decentralized pomdps. In *Proceedings of the Sixth European Workshop on Reinforcement Learning*.

Bernstein, D.; Givan, R.; Immerman, N.; and Zilberstein, S. 2002. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research* 27(4):819–840.

Boutilier, C. 1999. Sequential optimality and coordination in multiagent systems. In *Proceedings of the Sixteenth International Joint Conferences on Artificial Intelligence (IJCAI-99)*.

Chades, I.; Scherrer, B.; and Charpillet, F. 2002. A heuristic approach for solving decentralized pomdp: Assessment on the pursuit problem. In *Proceedings of the Seventeenth ACM Symposium on Applied Computing (SAC-2002)*.

Decker, K. S., and Lesser, V. R. 1992. Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems*.

Decker, K. S., and Lesser, V. R. 1993. Quantitative modeling of complex computational task environments. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 214–217.

Goldman, C. V., and Zilberstein, S. 2003. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the Second International Conference on Autonomous Agents and Multi-agent Systems*.

Jennings, N. R. 1996. Coordination techniques for distributed artificial intelligence. In O'Hare, G., and Jennings, N., eds., *Foundations of Distributed Artificial Intelligence*. John Wiley.

Kirn, S. 1996. Organizational intelligence and distributed artificial intelligence. In O'Hare, G., and Jennings, N., eds., *Foundations of Distributed Artificial Intelligence*. John Wiley.

Nair, R.; Tambe, M.; Yokoo, M.; Pynadath, D.; and Marsella, S. 2003. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*.

Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley.

Pynadath, D., and Tambe, M. 2002. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *JAIR* 16:389–423.

Wagner, T. A.; Garvey, A. J.; and Lesser, V. R. 1998. Criteria directed task scheduling. *Journal for Approximate Reasoning: Special Scheduling Issue* 19:91–118.

Xuan, P., and Lesser, V. 1999. Incorporating uncertainty in agent commitments. In *Intelligent Agents VI: Agents, Theories, Architectures and Languages (ATAL), Proceedings of The Sixth International Workshop on Agent Theories, Architectures, and Languages (ATAL-99), Lecture Notes in Artificial Intelligence 1757*. Springer-Verlag.

Xuan, P.; Lesser, V.; and Zilberstein, S. 2001. Communication decisions in multi-agent cooperation: Model and experiments. In *Proceedings of the Fifth International Conference on Autonomous Agent (AGENTS 01)*, 616–623.