AI Magazine Volume 20 Number 1 (1999) (© AAAI)

Practically Coordinating

Edmund H. Durfee

■ To coordinate, intelligent agents might need to know something about themselves, about each other, about how others view themselves and others, about how others think others view themselves and others, and so on. Taken to an extreme, the amount of knowledge an agent might possess to coordinate its interactions with others might outstrip the agent's limited reasoning capacity (its available time, memory, and so on). Much of the work in studying and building multiagent systems has thus been devoted to developing practical techniques for achieving coordination, typically by limiting the knowledge available to, or necessary for, agents. This article categorizes techniques for keeping agents suitably ignorant so that they can practically coordinate and gives a selective survey of examples of these techniques for illustration.

It has been said that ignorance is bliss. Certainly, people who know much (or think they know much) are sometimes subject to cockiness, confusion, paralysis, resignation, or other unpleasant states. Artificial agents can also suffer from knowing too much, and so, it behooves us, as agent designers, to make sure that our agents are not overwhelmed with too much knowledge. The trick (or, ultimately, the engineering skill) is to design agents that have enough knowledge to act well in their environments, and no more knowledge than that, lest the knowledge over and above what is sufficient degrades the quality or timeliness of the actions.

The purpose of this article is to look more deeply at strategies for designing agents, and the strategies' methods of acquiring and using knowledge about agents, that make it practical for an agent to coordinate its actions with others. Before we begin, however, let me first be a little clearer about what I mean by the terms *agent* and *coordinate*. In keeping with much of the current use of the term, I consider an *agent* to be an entity that is capable of acting in its environment to satisfy its desires. Thus, an agent is an entity to which it is convenient to ascribe characteristics such as choices (capabilities for action); awareness (beliefs about the world); and preferences (over states of the world, often coming about as outcomes of its [and others'] actions). In general, the set of actions available to an agent, and the set of possible worlds, could be infinitely large. To keep matters simpler, however, I discretize them. We can capture these three characteristics in a variety of notations. For illustration purposes, let me here just represent them in terms of a payoff matrix, such as in figure 1. Here, agent *P* can take actions *A* or *B*, has beliefs about whether the world is in state of affairs (SOA) 1 or 2, and prefers to take actions that lead to higher payoffs (the numbers inside the matrix). For example, *P*'s actions might be to choose one of two doors, behind one of which is a lovely bouquet of flowers, the sight of which *P* values at 2 (figure 1). If *P* guesses the wrong door, he/she does not get the reward of seeing the flowers, although he/she is still just as able to enjoy their aroma, so he/she breaks even (a reward of 0).

When the outcomes of its choices can depend on the choices that other agents have made, are making, or will make, then an agent should consider the actions of these other agents when making its own choice. I refer to this—the taking into account the choices of others when deciding what to do—as an agent *coordinating* with these others. Coordination is an overloaded term, possibly meaning either the process or the result of coordinating. I use the former sense of the word, which has several advantages. For one thing, it removes value judgments that go along with evaluating whether a result is coordinated. When such evaluations are needed to convey a sense of agents working to their mutual benefit, for example, I prefer a term such as *cooperative* rather than *coordinated*. By considering coordination as the process by which an agent takes into account the possible actions of others (uses these actions as "coordinates" to index

P \ SOA	1	2
A	2	0
B	0	2

Figure 1. A Simple Decision Situation.

Agent *P* has a choice between doing *A* or *B*. The reward it receives is dependent on the state of affairs (SOA).

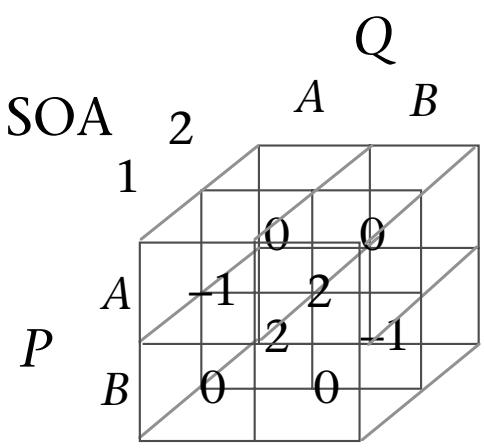

Figure 2. Coordination Problem.

The payoff to *P* depends on its choice, the choice of *Q*, and the state of affairs (SOA). To decide on its choice, it should use whatever it knows to anticipate the choice of *Q* and, thus, coordinate with this choice.

into an outcome), we can determine whether an agent has coordinated without looking at other agents. Because coordination does not imply any mutuality, cooperative or competitive, it need not even be symmetric!

Returning to our simple example, what if *P* shared its world with *Q*? Now, it turns out that if *P* and *Q* choose the same door, the object (if any) behind the door is removed. Thus, if they both choose the door with the flowers, the flowers are removed, and *P* is actually worse off (reward of –1), now being unable even to smell the flowers any longer. If *Q* opens the door with the flowers, then *P*'s view is obstructed; so, *P* is no better or no worse off. From the perspective of *P*, therefore, it might be better to consider *Q*'s likely choice when making its own decision, as shown in figure 2, because *P* can benefit only when it chooses a different action than *Q*!

To make its decision, *P* needs to determine whether the state of affairs is 1 or 2 and whether *Q* is likely to do *A* or *B* and then *P* should act accordingly. The degree of detail with which *P* should model *Q* would thus depend on how much *P* needs to know about *Q* to make an adequate prediction about *Q*'s choice. Perhaps it is enough for *P* to have a probability associated with *Q* doing each of its actions. If *Q* were a degenerate agent, such as an agent that simply flips a coin to choose an action, then this degree of modeling detail might suffice. If *Q*'s actions, however, are more dependent on the situation, then perhaps *P* needs probabilities for *Q*'s actions conditioned on the state of affairs (SOA). For example, perhaps the aroma of flowers wafting from one of the rooms is quite discernible to *P*, and *P* has some statistics of how often *Q* takes action *A* versus *B* when *P* has sensed the aroma from this room before.

Of course, rather than relying on past statistics, *P* could use what it knows about *Q*'s preferences. For example, if *P* thinks that *Q* likes flowers too (that is, *P* thinks that *Q*'s preferences are just like *P*'s in figure 1), then *P* might conclude that *Q* will choose door *A* in SOA 1 and *B* in SOA 2. *P* can use this conclusion to coordinate its decision with that of *Q*. It is also possible that *P* and *Q* might have different beliefs about the state of affairs (perhaps *P* thinks *Q* has a cold and cannot smell very well); so, to predict *Q*'s likely action, *P* really should attempt to infer *Q*'s probability distribution over the states of affairs.

P could even believe that *Q* will consider *P* when making its own decision. Thus, *P* will need to model what it thinks *Q*'s model of *P* is. To decide what to do, therefore, *P* will need to determine what it thinks *Q* will do, which, in turn, requires that *P* determines what it thinks *Q* will think that *P* will do, which, in turn, could require that *P* determines what it thinks *Q* thinks that *P* will think that *Q* will do, and so on. In principle, such nested models that

agents have of each other could continue indefinitely.

Recursive Modeling Method

What should an agent do in such circumstances? Well, one answer is that it should use everything that it knows to make a good decision. Using all its knowledge means being able to represent the knowledge and process it. For example, let us say that *P* clearly smells the flowers behind the door that would be opened with action *A*; so, the decision it faces can be reduced to the matrix at the top of figure 3. *P* is also certain that *Q* can smell the flowers and that *Q* values outcomes as does *P*; so, *P* models *Q*'s decision situation at the second level of figure 3. However, *P* also believes that *Q* thinks *P* cannot smell the flowers (*P* only recently got over a cold, so does not expect *Q* to know this, for example); so, *P* thinks *Q* will think *P*'s choice will amount to randomly picking an action, represented at the bottom level of figure 3 as (probability of choosing *A*, probability of choosing *B*).

A dynamic programming strategy, as used in the recursive modeling method (RMM) (Gmytrasiewicz and Durfee 1995), can solve this decision problem by propagating from the leaves upward. *P* will believe that *Q* will take action *A* (because with *P* acting randomly, *Q* will expect an average payoff of 1/2 for action *A* and 0 for action *B*); so, *P* would maximize its expected payoff by taking action *B* (with payoff 0, compared to expected payoff of –1 for action *A*).

Somehow, this outcome seems unsatisfactory. After all, *P* seemingly knows more about Q, yet *Q* is likely to get the higher payoff. It is to *Q*'s advantage that it is seen as ignorant by *P*![1] If *P* were to ignore its beliefs in *Q*'s ignorance, then it might be more inclined to take action *A*, but of course, if *P*'s model is accurate, any increased probability in its taking action *A* only decreases *P*'s expected payoff. It also drags *Q*'s payoff down with it, but we assume that *P* would not care about *Q*'s payoff. (If misery loves company, then *P* might want to reduce *Q*'s payoff, but this implies that there is more to *P*'s perceived payoffs that should be represented in its matrix.)

P might be able to improve its position if it were to (justifiably) change its knowledge state. For example, it might tell *Q* that it is now able to smell the flowers, which means that its model of *Q*'s model of *P* should change. Now *P*'s model of *Q*'s model of *P* might include *P*'s payoff matrix, which, in turn, means that *P* would have a model of *Q*'s model of *P*'s model of *Q* (at least) to predict how *Q* will think *P* will react to what it thinks *Q* will do! Obviously, the deeper *P*'s knowledge about *Q*'s knowledge about *P*'s knowledge about…the more extensive the representation and computation are.

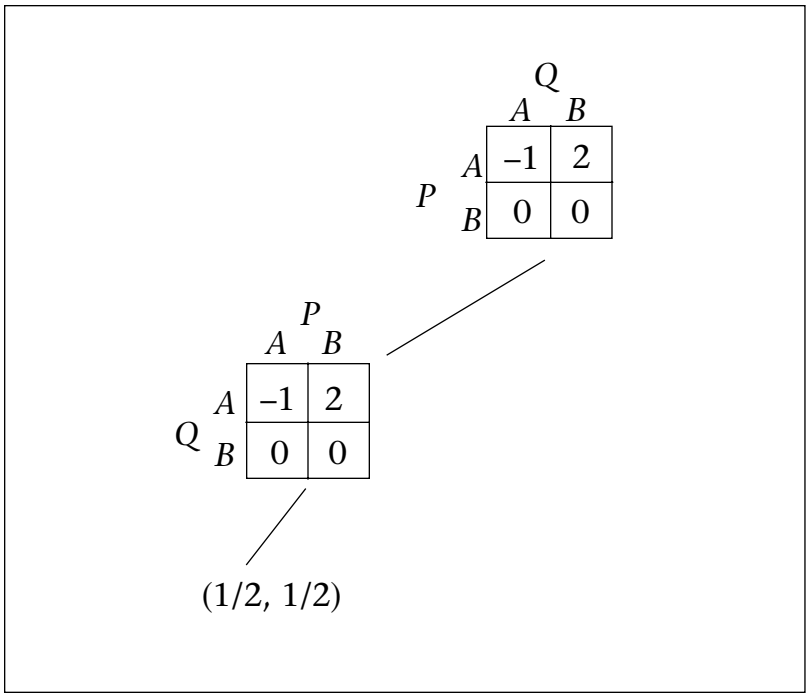

Figure 3. An Example Recursive Modeling Method Hierarchy.

P models its decision situation (at the top), the situation it believes that *Q* is facing (in the middle), and the knowledge that it thinks that *Q* has of *P* (at the bottom). This last knowledge indicates that *P* thinks that *Q* thinks that *P* is equally likely to do *A* or *B*.

In addition, the size of the nested models might not only be the result of the depth of knowledge but also the breadth of possibilities. For example, what if some agents in the world were allergic to flowers? For such agents, the best outcome is for both agents to choose the door with the flowers, so that the flowers are removed from the vicinity! Perhaps *P* is uncertain about whether *Q* is allergic or not and whether *Q* will think that *P* is allergic or not, and perhaps *P* even has beliefs deeper than that, yielding a representation such as the one in figure 4.

The uncertainty *P* has (or thinks *Q* has, or thinks *Q* thinks *P* has,…) is represented in the branches. *P* believes that *Q* is not allergic to flowers with probability p_1. *P* believes that if *Q* is not allergic, then *Q* will believe *P* is not allergic with probability p_2, but if *Q* is allergic, then it will believe *P* is not allergic with the

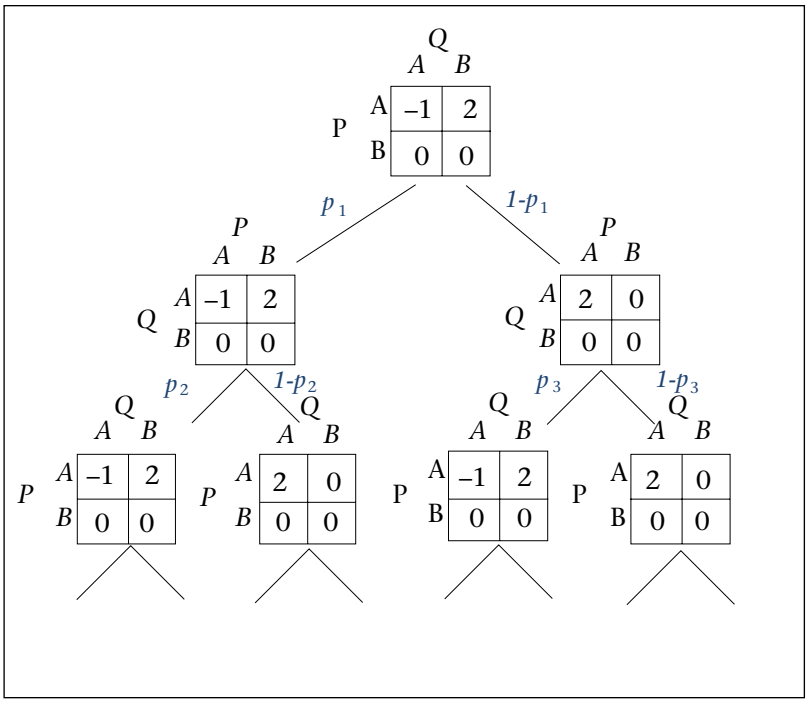

Figure 4. A Branchy Nested Model.

In making its decision (top), *P* considers it possible that *Q* could see the situation in either of two ways (second level) and that for each of these *Q* will believe *P* could see the situation in either of two ways (third level), and so on.

possibly different probability p_3, and so on. In general, the branching factor can be much larger than 2. Moreover, the contents of the payoff matrixes and the values of the probabilities can vary in principle, limited only by the knowledge available to an agent (because the hierarchy summarizes an agent's coordination knowledge). Although in theory using all the knowledge it has or might get will always lead to better (or at least no worse) decisions for an agent,[2] the costs of acquiring and using the knowledge must be considered to make the approach practical.

Keeping Coordination Practical

Using RMM as an example, consider the amount of reasoning that an agent might have to do. To consider each of the combinations of choices that the agents have, an agent needs to identify the choices (actions, plans) for each of the agents and the outcomes (utilities) of each combination of choices for each of the possible environments. If we assume each of the n agents has c choices, then there are cn choices to identify for an environment. Because each choice can correspond to a planned course of action on the part of an agent, there are cn plans to formulate. In addition, for each of the c^n combinations of plans, the outcome(s) of executing the plan combination must be predicted and assigned values (utilities). The previous calculations only correspond to one view of the interaction by an agent. Given nested views of how an agent thinks that others think…that others think about the interaction, there could be b^l such models to construct, where b is the branching factor caused by uncertainty (such as about allergies in the running example), and l is the depth of the nesting of models available to the agent.

As an agent comes to know more, it must in general do exponentially more computation. Because all practical agents have limits to the resources they can apply to make coordination decisions, it is in an agent's (and an agent designer's) best interests to maintain as much ignorance about the world and the agents that populate it as it can, while knowing enough to coordinate acceptably well with others. If we consider all the possible knowledge, as outlined within the RMM framework, there are numerous places where we could hope to trim the knowledge being used (figure 5). We can be selective about the nested knowledge we use or even obviate its use by exploiting communication (bottom of the figure). We can simplify the utility calculations, trim the number of options evaluated for each agent, or decrease the frequency of coordination decision making by coordinating over longer-lived choices (middle of figure). We can even reduce the dimensionality of an interaction by ignoring agents or viewing groups of agents as individuals (top of figure). In short, by considering places where an agent can simplify its coordination task by being selectively ignorant, we can make coordination practical. In the remainder of this article, I examine these strategies for practical coordination in more detail, considering examples of such methods. To provide a framework for the discussion, I work from the bottom of figure 5 upward.

Limited Use of Nested Models

One method of keeping the computation in check is to prune away portions of the nested models, a technique familiar in minimax algorithms for game-playing programs. Pruning nested knowledge is somewhat different from game-playing reasoning, however, because unlike game-playing reasoning, which hypothesizes sequential possible physical game states, the nested knowledge captures what amounts to simultaneous perspectives on the parts of the agents. Thus, in game playing,

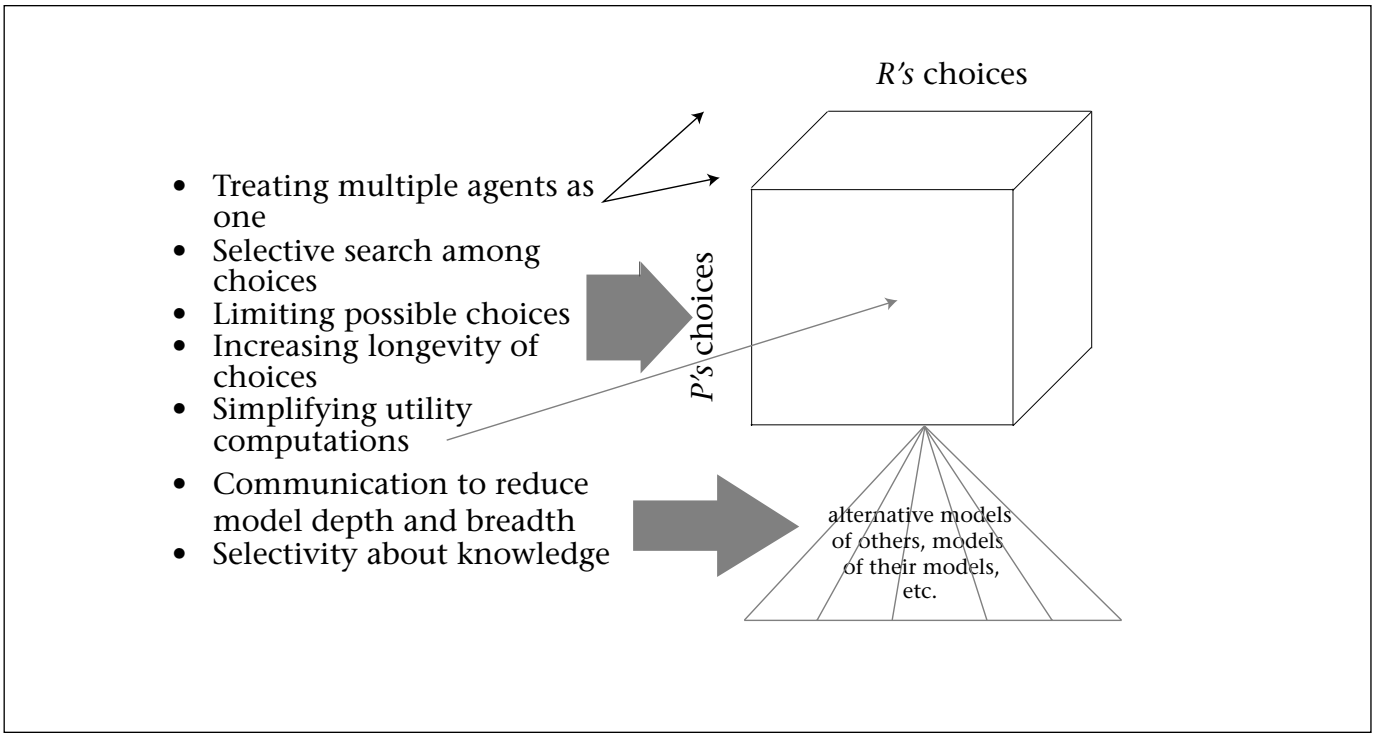

Figure 5. Strategies for Making Coordination Practical.
The outcome of *P*'s actions depend on the actions of the other agents (*Q* and *R* in this figure, but in general this is *n*-dimensional). Anticipating their actions can require nested models of how they see the situation, how they think others see the situation, how they think others think that others see the situation, and so on. Making this reasoning tractable means limiting the number of nested models to reason over and decreasing the number of action combinations that *P* must consider.

undesirable states can be pruned because rational agents will not act to get into these states. With nested models, however, the models exist regardless of their desirability; ignoring an unpleasant fact will not make it go away. For example, in the case where *P* believes *Q* does not know that *P* can smell the flowers, *P* might choose to ignore what it knows about *Q*, treating *Q* as equally likely to take either action (which is a natural way of terminating the recursive nesting when an agent is seen to have no information at a deeper level). If Q were equally likely to take either action, then *P* would take action *A* with an expected payoff of 1/2. However, in reality, action *A* will give *P* a payoff of –1, and *P* knew this to be the case but chose to ignore it.

The strategy, therefore, is to prune away possible nested knowledge that is not expected to change the strategy choice of the agent doing the reasoning (Russell and Wefald 1991) rather than prune undesirable states as such. For example, in figure 4 at the second level right-hand side, *Q*, if allergic, has a weakly dominating strategy of taking action *A* (the only way it would ever do *B* is if it were convinced that *A* was going to do *B*, and even then *Q* would be indifferent between *A* and *B*). Thus, *P* might choose to not search deeper in the branch because it is unlikely that anything it discovers down there will change what it will expect *allergic*-*Q* to do. Our preliminary results using such a strategy indicate that this approach holds promise, even for simple application domains. Vidal and Durfee (1996, 1995) show this strategy's promise in a simple pursuit task, where four predator agents placed in a grid world must surround a prey. Given an uneven distribution of predators, they have to decide which will block the prey from which direction. An individual predator will thus seek to occupy the closest unoccupied side but to decide which sides are likely candidates it needs to determine the sides that other predators are likely to occupy. However, because this determination, in turn, depends on what sides they think others will occupy, and so on, the nesting can be fairly complicated. By using heuristic estimates (based on previous experience) about the impact of expanding different parts of the nested models, the agents in this problem domain can achieve a high level of cooperative behavior using only a judiciously chosen subset of the nested models.

Using Regularities in Nested Models

Another way of simplifying the nested model computations is to take advantage of patterns in the nested structure to avoid rederiving the same information in different places. Figure 4 illustrates how regularities in knowledge can lead to patterns of similar models. Of course, if the knowledge is finite, then eventually some patterns must be broken. However, a powerful simplifying assumption can be to purposely project the models beyond finitely available knowledge to infinite levels.[3] When this projection is done, fix-point solutions can be discovered using often much simpler computational means. The use of fix-point solutions has been a standard approach in game theory, for example, in identifying equilibrium solutions. Using the simple view of figure 3 as an example, *P* might simplify its model by assuming that it and *Q* model each other identically and that this is common knowledge (that is, that *P* knows that *Q* knows that *P* knows that *Q* knows that…*P* and *Q* model each other identically, where the…can go on infinitely deeply). Then, in the world where *P* smells and wants to see the flowers, it models *Q* as doing the same, models *Q* as thinking *P* will do the same, and so on. The symmetric nature of this situation leads to *P* believing that it and *Q* will come up with identical strategies. In this case, for example, *P* might conclude that it and *Q* will have mixed strategies of taking action *A* with probability 1/3 and *B* with probability 2/3, yielding an expected payoff to each of 1/3.

An extremely common means for transforming knowledge states, …, is to change the knowledge that agents have through observation.

Exploiting Observations

To this point, we have considered how an agent, with its particular nested models about the coordination situation, can make its coordination reasoning more practical by transforming its nested knowledge into an approximate form that might require less reasoning effort. An extremely common means for transforming knowledge states, however, is to change the knowledge that agents have through observation.

Observations, for example, can be used for plan recognition. Based on the evidence provided by observations of another agent's actions, an agent can hypothesize the likely larger plans of which these actions are part (for example, Huber and Durfee [1996, 1995]). With such hypotheses, the agent can then anticipate the future actions of the other agent as those that continue the inferred plans. When actions and plans are sufficiently unambiguous, plan recognition can obviate the need for nested models.

Observations also provide evidence to learning mechanisms so that agents can learn what actions others are prone to take in various situations. The literature on this subject is growing rapidly (for example, Weiss and Sen [1996]). A fundamental challenge in this field is correlated with the notion of nested models; namely, although one agent is learning about others, they, in turn, are learning about it. That is, what an agent is learning is a moving target because every time it learns something that changes how it responds to some situation(s), its new responses can, in turn, lead others to respond differently to the situation(s). This learning by other agents, in turn, could lead to the agent learning to respond yet differently, and so on.

It is conceivable that in such circumstances, the learning never ends. In our running example, for instance, if *P* (who likes flowers) is paired repeatedly with a *Q* who is allergic in this game, we can intuitively picture *Q* seeking to match *P*'s choice (to eliminate the flowers from the world) while *P* seeks to make a different choice than *Q*. They could chase each other around the four combinations of actions indefinitely, as first one changes, then the other.

In general, a system reaches an equilibrium where the improvements it continues to make through learning are offset by the degradation to what it has already learned because of the volatility in what other agents are doing because of their learning. This notion is illustrated in figure 6, which plots the error rate of an agent at time t+1 given its error rate at time t, based on the combined learning capability (which tends to decrease error) and volatility (which tends to increase error). By characterizing learning capability and volatility (based on how coupled agents' actions are) for particular problems, we can build expectations of how practical it will be to use learning for coordination as well as whether the system as a whole can ever learn to eliminate all errors (Vidal and Durfee 1998).

Using Communication

Communication is commonly used to obviate the need for nested models. For example, one approach has been for agents to explicitly tell each other about their intended actions (or at least about constraints on what these actions might be). Such an approach has formed the backbone of work in multiagent planning, for example, where agents separately form their

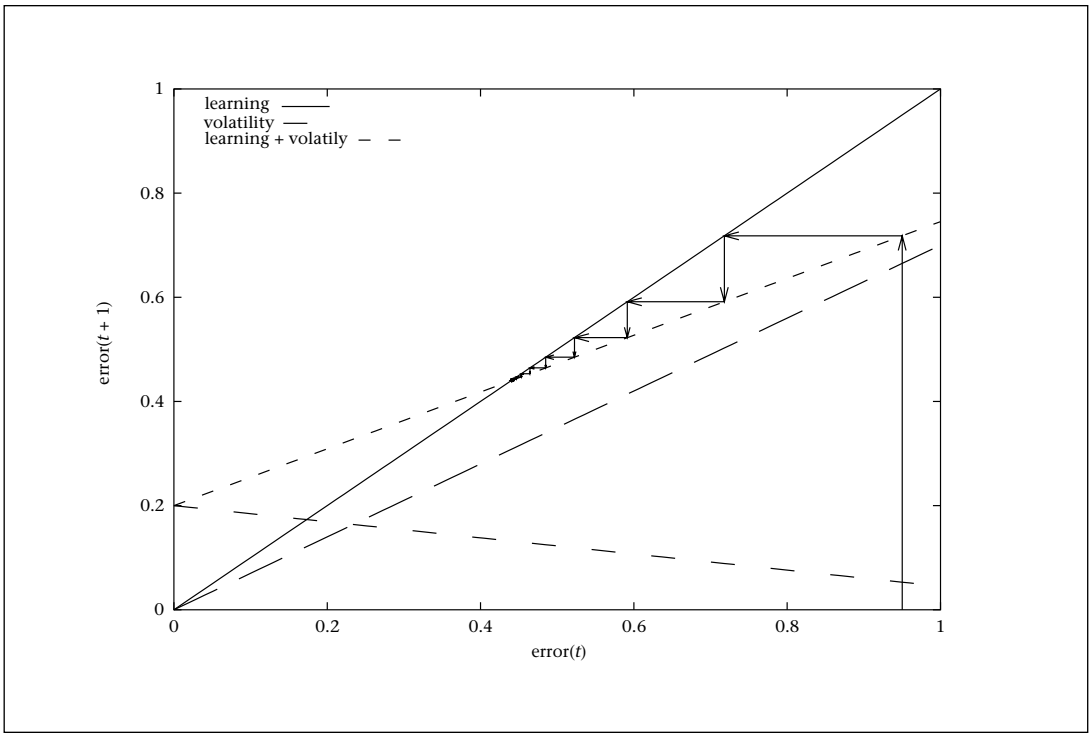

Figure 6. Error Progression.

The error rate at time t +1 based on the error rate at time t and the sum of the learning and volatility curves. The initially high error rate decreases but cannot fall below .44 because further learning by this agent is offset by the fact that learning by other agents renders obsolete some things that were previously learned.

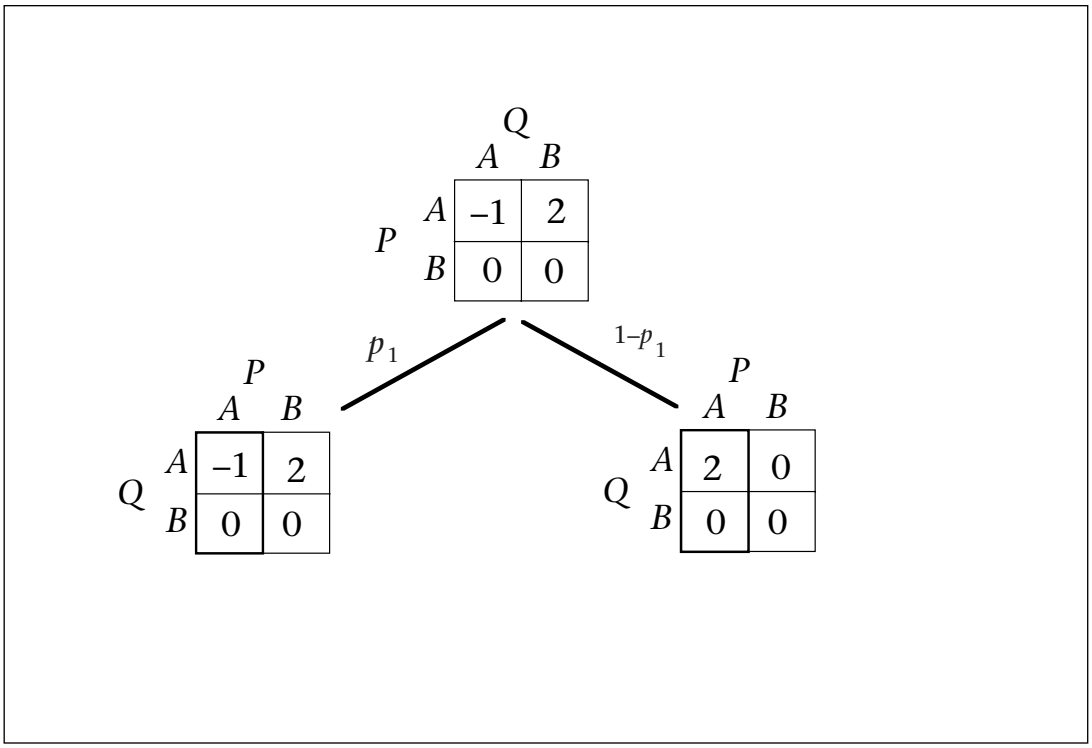

Figure 7. Truncated Hierarchy from Communication.

When *P* tells *Q* that it will do action *A, P* can now truncate its model because it need not reason about what *Q* will think *P* will do.

Often, …, designers of multiagent systems also want to be able to claim particular properties for the collective decisions of the agents, such as stability, efficiency, and fairness …. Thus, communication might be expected to change the agents' states of knowledge … to reach a state of knowledge from which certain properties must emerge.

own plans and then communicate to identify possible conflicts or cooperative opportunities (Ephrati and Rosenschein 1994; Durfee 1988; Georgeff 1983; Corkill 1979).

The benefits of such communication are clear when captured in a nested model framework such as RMM. By revealing information about itself, an agent simplifies its models of others, either by reducing uncertainty about the world (hence reducing the branching factor) or uncertainty about what others will be doing (hence reducing the requisite modeling depth). For example, if *P* considers telling *Q* "I will do *A*," then *P* would model the resultant mental situation as truncated because it knows exactly that *Q* will consequently expect *P* to pursue A (assuming that the message is certain to be delivered and believed [Gmytrasiewicz and Durfee 1993]).

Looking at the resultant model (figure 7), *P* can decide that its expected payoff in the knowledge state after sending the message is dependent on p_1. Specifically, it is $(3p_1-1)$. Clearly, being the first to commit to doing *A* is a good idea if *Q* is unlikely to be allergic and a bad idea if *Q* is allergic. With probabilities in between, the question of whether the message is good for *P* depends on what *P*'s expected kpayoff was before sending it and what it is afterward. *P* should send messages that cause the largest positive gains in its expected utility (Gmytrasiewicz, Durfee, and Wehe 1991).

Obviously, it might not be possible (or considered "fair") for *P* simply to claim action *A* by being first. Nonetheless, communication can still benefit the agents if they have correlated preferences. That is, even with some randomization thrown in about who gets first choice, they can still avoid mutually poor outcomes (such as both choosing *A* if neither is allergic). They could, for example, agree to abide by the flip of a fair coin such that each now has an expected utility of 1, which is better than 1/3, what they would expect to get without communication.[4] The case is even more obvious when both agents are allergic, where they both want to take the same action and get the same payoff! When possible, therefore, communication is often a practical and effective tool for the agent-coordination process!

Epistemic States for Well-Defined Coordination

In the previous subsection, we saw how a communicative act could be in the self-interest of an individual. Often, however, designers of multiagent systems also want to be able to claim particular properties for the collective decisions of the agents, such as stability, efficiency, and fairness (Durfee and Rosenschein 1994). Thus, communication might be expected to change the agents' states of knowledge, as we have seen, to reach a state of knowledge from which certain properties must emerge.

For example, a recent trend in game-theoretic research has been to revisit notions of decision-theoretic rationality as embodied in game-playing agents to understand better how rational agents would actually play a game, as opposed to analyzing a game independent of the agents, assuming basic rationality and common knowledge among the agents. Aumann and Brandenberger (1995), for example, investigate epistemic conditions for achieving Nash equilibria—that is, what must agents know if we are to be assured that their decisions will constitute a Nash equilibrium? Although their analyses are too involved to detail here, and introduce problematic notions of common knowledge for cases involving more than two agents, we can get a flavor of how their ideas dovetail into those of AI by considering our ongoing two-agent case.

Recall that in the previous subsection, *P* recognized that by telling *Q* about *P*'s intention to take action *A*, *P* could increase its own expected payoff. Moreover, in this case, *P* got this higher payoff by doing what it had told *Q* it would do (it did not need to lie). If *P* is correct in its knowledge of *Q*'s payoff matrixes, *Q*'s rationality, and *Q*'s correctly receiving *P*'s message, then *P* not only will take its part in a Nash equilibrium but, in fact, will know that the agents are in a Nash equilibrium if it has itself been rational in choosing its action and truthful in revealing it.

A challenge, of course, lies in some of these nested knowledge assumptions. For example, *P*'s projected knowledge state after sending the message (figure 7) is predicated on knowing that at the time of the door-opening decision, *Q* will have received, decoded, and incorporated *P*'s message. *P* could require that *Q* acknowledge the message, and this acknowledgment could suffice, although more interesting kinds of coordination requiring infinitely nested (common) knowledge might require infinitely many acknowledgments in principle (Fagin et al. 1995). Alternatively, *P* could have more models of *Q* (for the combinations of hear–not hear and allergic–not allergic) with their associated probabilities.

As we have seen, it is often advantageous to agents if they can attain (and possibly help others attain) particular states of knowledge. When particular kinds of knowledge state tend to be advantageous repeatedly, agents can dis-

cover patterns of communication that tend to lead to these states. These patterns of communication form the basis of protocols. Practical coordination that is predicated on communication usually embeds well-defined protocols into the agents to streamline the process of achieving knowledge states that are desirable for systemwide properties. Substantial efforts on the parts of multiagent system designers have gone into formulating, implementing, and testing such protocols (for example, KQML [Mayfield, Labrou, and Finin 1996; Cohen and Levesque, 1995]) so that agents reach agreement on issues such as task assignments (for example, Smith [1980], Rosenschein and Zlotkin [1994]) and coordinated plans (for example, Durfee [1988] and Ephrati and Rosenschein [1994]).

Finally, it should be mentioned that overuse of protocols can be counterproductive. A protocol that keeps agents informed about each other is generally helpful, but it could happen that when one agent changes its plans because of unexpected events, it tells others, who adjust their plans, and tells yet others who adjust their plans, and so on. Sometimes, this kind of chain reaction can trigger a large amount of communication and coordination reasoning; therefore, it only makes sense to inform others about a change in plans if the precipitating change was significant. Thus, part of the practical use of protocols is in deciding when it is better to say nothing at all (Durfee and Lesser 1988).

Constraining Choices

So far, we have focused on methods for keeping the nested modeling tractable by using internal reasoning, learning, or selective communication to reduce the depth or breadth of the modeling space. Even if we were to reduce these drastically, however, it would not help if what little remained involved huge interaction representations. That is, in the form we've been focusing on, reducing the number of nested matrixes will not help much if constructing even one such matrix is intractable. Recall that in the worst case, every possible combination of the c choices for the n agents must be evaluated, meaning c^n evaluations.

Certainly, if the space is to be explored exhaustively, the set of choices must be finite. By further constraining choices available to agents, we can simplify the representations of interactions. In the most degenerate case, where each agent is given a single specific capability, an interaction is a matrix with only a single element ($c = 1$ so $c^n = 1$)! This is an "assembly line" model of multiagent systems.

Organizational Structures

The idea of constraining the choices of agents has been part of multiagent systems for well over a decade in work that has tried to use notions of organizations and organizational roles as guidelines for coordinating agents (Corkill 1983). A description of an organization captures the preferences and responsibilities of its members, providing strong clues to the choices of actions that the members might make. The organization might also impose prohibitions on certain kinds of action, both for particular niches in the organization and all members of the organization.[5] By embedding agents within an organization, their decisions are simplified (they have fewer choices and know that others have fewer options), and their dynamic coordination activities can be better directed. Among the challenges in designing an organization is determining how to decompose tasks into separate roles to have reliable, inexpensive, and effective performance on the part of the organization.

For example, consider hierarchical organizations for tasks such as information gathering. Given a query to answer, the query task can be decomposed and the (independent) query subtasks assigned. Let's define the task granularity (γ) as the task-execution time (τ) divided by the communication delay (δ). Then, given γ, the total number of primitive information-gathering tasks (N), and the number of tasks (m) assigned to each leaf of the organization, we can derive the branching factor k for the balanced tree that minimizes response time (So and Durfee 1996):

$$T(N,k,\gamma,m) = \begin{cases} (k+1)/\delta + (l+m)\tau \rightarrow \gamma \leq 1 \\ 2/\delta + (kl+m)\tau \rightarrow \gamma > 1 \end{cases}$$

In the previous equation, $l = log_k (N/m)$ and is the number of levels in the organization. The processing at the leaf level is simply to execute the primitive tasks and send the results of each back up the hierarchy. At the nonleaf levels, the agents receive larger tasks from above, break them into k subtasks, assign these subtasks sequentially to the k agents below; then as results are returned, they are synthesized (where integrating a received result requires τ time), and the composite result is sent back up the hierarchy once it is done.

Under this assumed organizational behavior, it is the case that for $N = 32$, $m = 2$, and any γ, $k = 4$ outperforms $k = 2$ and $k = 16$. Even in the space of such simple hierarchical organizations as these, therefore, the detailed design of

Certainly, if the space is to be explored exhaustively, the set of choices must be finite. By further constraining choices available to agents, we can simplify the representations of interactions.

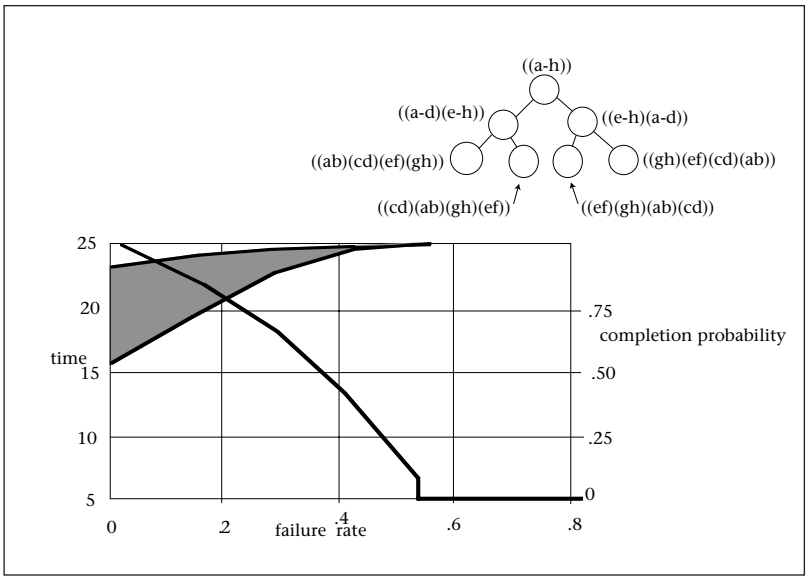

Figure 8. Binary Tree Organization.

The organization (upper right) has 8 tasks overall ($N = 8$), each nonleaf has 2 children ($k = 2$), each leaf does 2 tasks ($m = 2$), and tasks are assigned so that even if 1 of every pair of children fails, all the tasks still get done ($o = 1$). The graph shows how variance in execution time decreases with greater agent failure rates and how the probability of successful overall task completion decreases (scale on right side of graph).

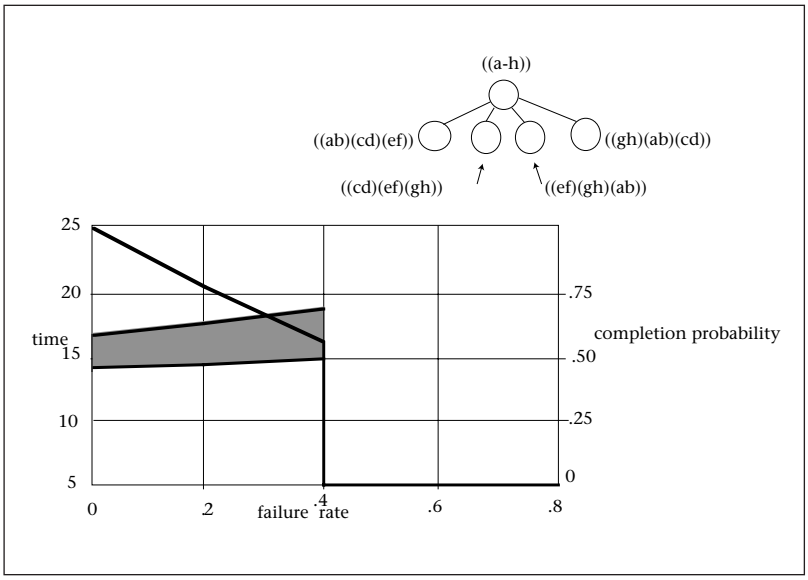

Figure 9. Flat Tree Organization.

The organization (upper right) has 8 tasks overall ($N = 8$), the 1 nonleaf has 4 children ($k = 4$), each leaf does 2 tasks ($m = 2$), and tasks are assigned so that even if 2 of the 4 children fails, all the tasks still get done ($o = 2$). The graph shows how variance in execution time decreases with greater agent failure rates and how the probability of successful overall task completion decreases (scale on right side of graph).

the organization (the selection of parameter values for features such as the number of subordinates for each manager) balances several considerations.

Organization and Run-Time Coordination Codesign

The previous organization analysis assumes that each agent can reliably accomplish its task(s). A major challenge in organization design, in fact, is to design reliable organizations that can tolerate failures of some agents. To increase reliability, we typically introduce redundancy among the agents so that each task is replicated by several agents. To the degree of replication, the task can still get done even if some agents fail. However, redundancy also opens the door to possible inefficiencies because agents can duplicate each others' work in situations with few or no agent failures. Duplication of effort can be avoided if agents are able to coordinate dynamically at run time, but this run-time coordination, in turn, incurs overhead and assumes sophisticated agents. Thus, an important question in organization design is, How do different organizations make demands on the sophistication of agents that populate them?

To begin answering this question, we have defined *o-redundancy task assignment* by a parent to a child in a treelike organization as an assignment that tolerates the failure of o children. For example, one-redundancy task assignment means that the organization is still assured of succeeding even if one of the children agents of each of the nonleaf agents fails. In a binary organization ($k = 2$), one-redundant task assignment means that all the agents are responsible for all the tasks because in the worst case (when only one of every two children survives), there is a single "live" path to a single leaf! Because the failures are random, every single such path must lead to success, so every leaf agent will have every subtask. (See figure 8 with tasks labeled with letters *a, b, c*…. Note that tasks get grouped into subsets at deeper levels, where the ordering of subsets might imply a preferred ordering of subtask execution.) Fortunately, for larger values of k, redundancy need not be quite so total! (See figure 9, where a leaf agent does not have all the subtasks.)

For different levels of o-redundancy, along with the previously described parameters, we can measure an organization's response time given that particular agents fail. We here define the agent failure rate as the proportion of agents that fail (although a richer model of independent failures is used elsewhere [Durfee and So 1997]). Thus, with a failure rate of .4, an organization with 5 agents will have 2 failed agents. Whether the organization succeeds can depend on which of the agents fail (note that any simple hierarchical organization such as the ones

we have examined will fail if the top agent fails), and in some cases, the failure rate might force the violation of the degree of redundancy in the organization, so the organization will be assured of failing to respond at all.

Thus, a particular organization will not have a deterministic response time but, rather, will have a distribution over performance times, depending on which agents fail and how the surviving agents order their tasks. If we assume random task ordering, we get behavior as exemplified in figure 8 and figure 9. (These figures do not include response times for failed instances of organizations [Durfee and So 1997].) Note how, in the first case (figure 8), the distribution over run times (shaded region) narrows with increasing failure rate. This narrowing is because, as more agents fail, the differences in performance as a result of alternative orderings disappears because ultimately, enough agents are gone that the remaining ones must complete all their tasks anyway. This narrowing does not occur in the second case (figure 9); having less redundancy at the leaf agents means the performance distribution will be narrower, but it tends to widen with failures because there can be more reliance on results from agents that get their tasks later (assignments are made left to right). Broader distributions represent opportunities for improvements through run-time coordination (Durfee and So 1997), which helps agents coordinate their orderings to do better than random. The second case also has a higher probability of completion for low agent failure rates but does not degrade as gracefully as the first case when agent failure rate increases, as shown by the solid line in the graph giving completion probability (scale on the right side of the graphs).

The design of organizations is, thus, a balancing act between factors such as reliability, response time, and investment in run-time coordination technologies for the agents that populate the organization (Durfee and So 1997). Again, recall that by embedding agents within an organization, their decisions are simplified (they have fewer choices and know that others have fewer options), and their dynamic coordination activities can be better directed.

Preference Simplification and Selective Search

As we have seen, overly constraining choices will affect the flexibility of agents to accommodate less than ideal circumstances, such as the failure of some agents within an organization. There are, therefore, limits to how much we want to make coordination practical by tying the hands of our agents—by keeping c small so that c^n stays small.

In this section, I consider alternative strategies, such as only selectively examining some of the c^n combinations or even just keeping n small. If some agents can safely be ignored, for example, the interaction can be simplified greatly. To say that an agent can be ignored means that the choices made by the agent have no (or negligible) impact on the perceived payoff that another gets from its choice of actions. Obviously, one way of realizing this negligible impact is to structure the multiagent system in a way that maximizes independence, such as creating organizations with independent roles (for example, no redundancy) so that agents would not need to consider what others had or would be doing when making their own decisions. Such systems have been called completely accurate, independent systems (Lesser and Corkill 1981).

In more open systems, imposing such structure can be problematic, so alternative means are needed. One fundamental approach is to simplify the agents' preference structure. Consider, for example, the following: In a robot delivery task, a robot R is indifferent to where other robots are, except when they are trying to be in the same place as R at the same time. Thus, if we consider R's interaction space, for most of the choice combinations, the payoff of the choice combination is the same as R's payoff for its individual choice, except for a few points where R's payoff is strongly negative (a collision!). Thus, reasoning about its choice of action could be viewed as prohibiting the bad combinations and acting independently otherwise. The notion of social laws (Shoham and Tennenholtz 1995), for example, has this flavor of prohibiting actions that lead to failure states and otherwise allowing agents to ignore each other.

Satisficing

An even greater simplification can arise if each agent has only two levels of preference over outcomes of choices: (1) good and (2) no good. In effect, this simplification reduces the search for an optimal choice of action to a satisficing search: As soon as a choice is found that is good, it is pointless to enumerate and evaluate other choices.

These strategies presuppose that agents know what choices to avoid. In some cases, offline analyses of an application domain can yield a set of prohibitions, as has been done with social laws (Shoham and Tennenholtz

… overly constraining choices will affect the flexibility of agents to accommodate less than ideal circumstances, such as the failure of some agents within an organization. There are … limits to how much we want to make coordination practical by tying the hands of our agents….

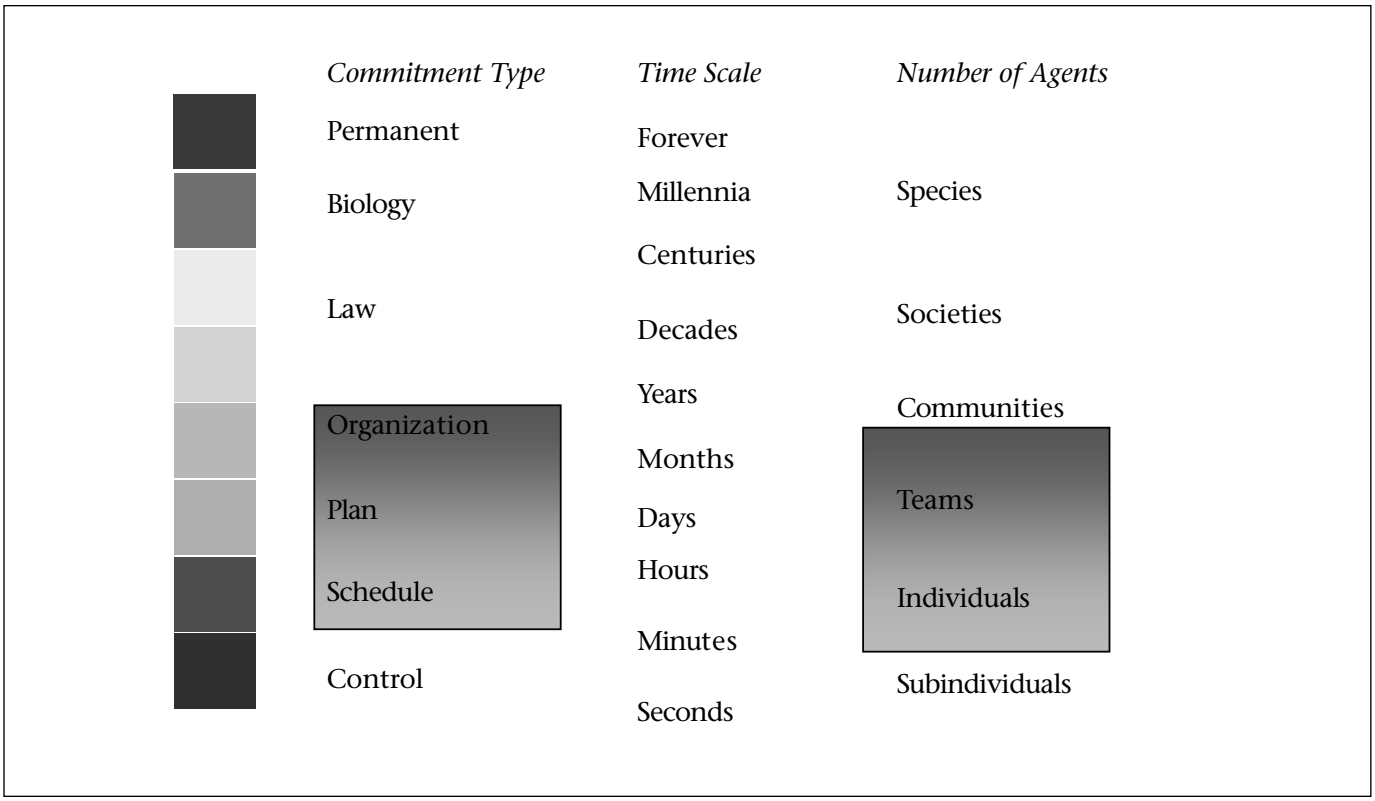

Figure 10. Commitment Spectrum.

As we work our way down, the time scale of commitments decreases, as does the number of agents participating in the commitment. The marked areas are the parts of commitment space focused on in this article.

1995). However, run-time methods for searching for coordinated choices can exploit simplified preferences greatly. For example, in coordination approaches based on plan merging (for example, Ephrati and Rosenschein [1994]; Durfee and Lesser [1991]; Georgeff [1983]), agents begin by assuming that their choices are independent; thus, each agent searches for plans that look best locally. These are then merged to detect conflicts, and if conflicts are found, some agents might replan or simply revise (such as insert synchronization actions into) their original plans to remove the problem. Thus, rather than enumerating the whole interaction space (matrix), the agents selectively enumerate portions of the space until a satisfactory combination of local plans is found. Often, they perform a hill-climbing search, beginning from their combination of independently derived local plans and searching through successive perturbations of these plans until a satisfactory (conflict-free) combination is found (for example, Durfee and Lesser [1991]).

This satisficing simplification of preferences to being good (enough) and no good reduces the coordination process to a distributed constraint-satisfaction problem (Yokoo et al. 1992; Conry et al. 1991; Sycara et al. 1991), where all satisfactory solutions are equally good. Much work in multiagent systems has benefited from this kind of simplification, taking advantage of algorithms for constraint-satisfaction search to solve, for example, problems in distributed resource allocation and scheduling. There might be many possible strategies for conducting such a search, however, and the quality and cost of coordination might well depend on adopting an appropriate strategy given the current problem-solving context.

Negotiation

For example, in the application domain of distributed meeting scheduling, there could be several strategies for how agents go about searching through (negotiating over) possible meeting times to propose to each other (Sen and Durfee 1995). Two (of many) possible strategies are to (1) simply move through the available times chronologically and schedule a meeting as early as possible or (2) find larger portions of the calendar that are relatively free and then iteratively narrow down the times to find a meeting time. These strategies lead to

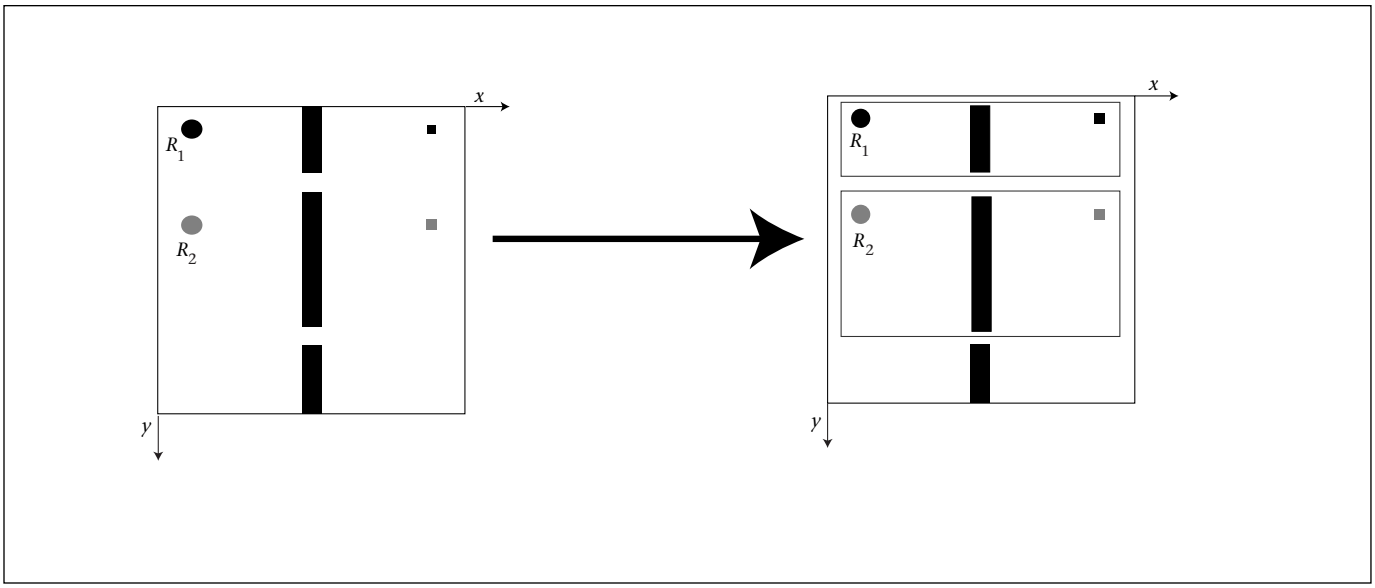

Figure 11. An Organizational Solution.

*R*1 delivers objects between the solid square locations (it is at one of those in the figure). R2 similarly delivers between the shaded regions. One coordination decision could be to partition the space once and for all, such that each robot has complete control over its region (shown on the right).

calendars that look different, the second tending to distribute meetings more evenly. In turn, the evolution of a calendar with one of the strategies eventually reaches a point where the other strategy becomes the better (more cost-effective) choice for further scheduling. As the calendar gets full toward the front from as-soon-as-possible scheduling, a fit-in-sparse-space strategy works better. Because the fit-in-sparse-space strategy tends to make the calendar evenly dense (so there really are no spaces appreciably more sparse than others), the simpler soon-as-possible strategy eventually becomes more cost effective. Thus, not only does an agent's choice of strategy for searching the options affect the quality and cost of its schedule, but the agent must also be capable of adapting its strategy as circumstances change.

The iterative search through the space of joint decisions that we have described has many of the features that most people associate with the concept of negotiation. There are many possible strategies for making negotiation practical depending on the needs of an application, ranging from simplifying the preferences and adapting the search strategy (as outlined earlier), to simplifying the proposals (for example, using prices to summarize agents' allocation plans, as in Wellman [1993] and Lee and Durfee [1995]), to using heuristics or past cases to generate new proposals based on feedback about prior proposals (as in Sycara [1989]), to exchanging intervals of proposals and narrowing down to solutions, and so on. In fact, because the term *negotiation* has been used to encompass so many more specific strategies such as those just mentioned, the term has become much less technically meaningful; a challenge for the community is to more carefully characterize the different kinds of negotiation that have been studied, possibly (as suggested in this article) by focusing on how each kind attempts to make the coordination problem tractable.

Hierarchical Elaboration of Choices

To this point, we have discussed approaches where agents can search through the space of individual actions to find good joint actions, such as working their way through proposed meeting times and agents working within longer-term organizational guidelines that focus their behaviors. In fact, these strategies for coordination involve agents making commitments at different levels (at the schedule level versus the organization level). Generally speaking, agents can make commitments along a spectrum of levels, which are differentiated mostly in terms of the lifetime of the commitment (or the frequency with which new commitments must be made) and in terms of the number of agents involved in the commitment, as broadly summarized in figure 10. The frequency of coordination decisions is

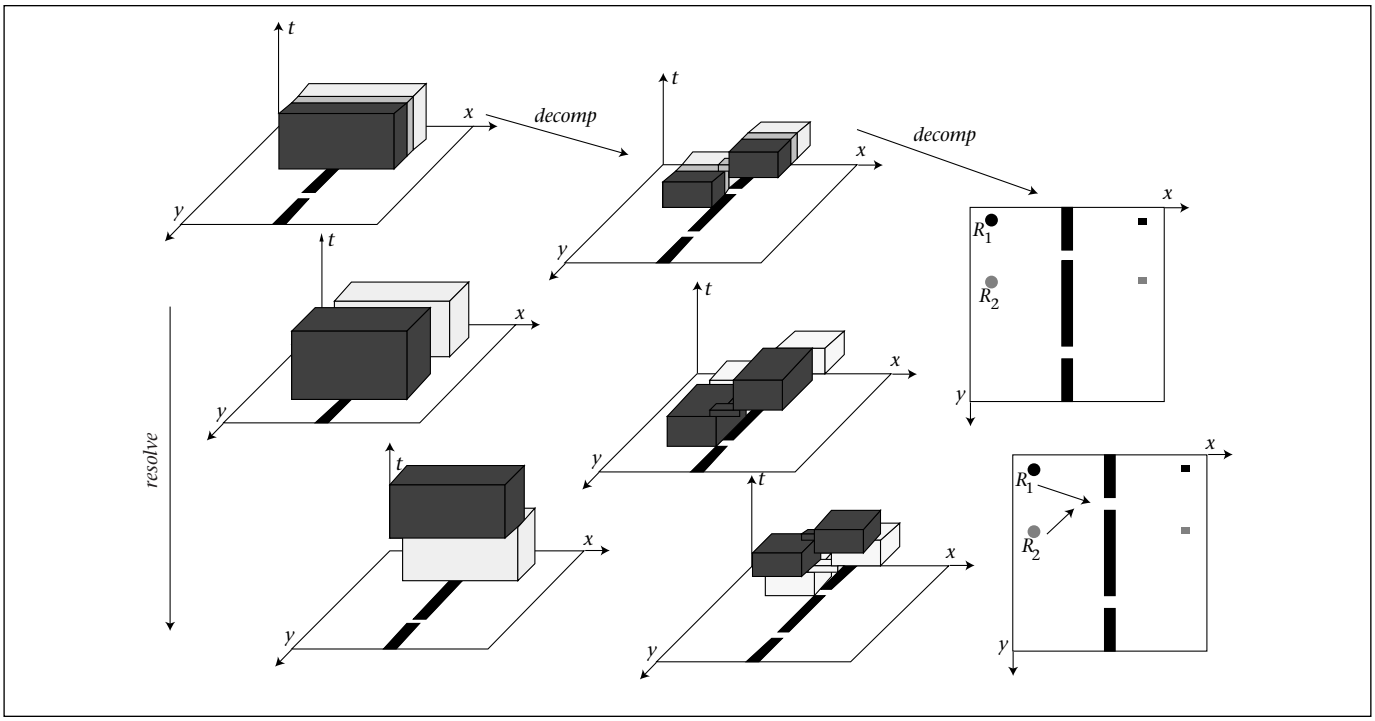

Figure 12. Alternative Levels of Abstraction.

Treating coordination for only the next delivery, the agents represent their plans abstractly in terms of an *x*-*y* region over time. The most abstract representations (upper left) overlap, leading either to moving their activities apart in space or time (working downward in the figure) or exchanging more detailed views (moving to the right in the figure) to isolate more precisely where the conflicts could arise. At the most extreme right, the agents exchange detailed movement plans and coordinate by shifting one of them very slightly in time to avoid collision in the doorway.

conveyed in the left column (darker is more frequent) along with the type of commitment and the time scale and number of agents involved. The emphasis in multiagent systems research is mostly in the areas that are boxed, with the gray areas being less fully explored. The choice of what kinds of commitment agent should make to each other depends on the frequency of coordination activity, the requirements for coordination precision, the tolerance of coordination costs, and the flexibility that agents need to individually retain to cope with environmental changes.

An ongoing objective of our work is to represent the continuous spectrum of commitments in a single search space to allow agents to move among models of individual and joint activity at different levels of (temporal) abstraction. Thus, the search for coordinated activity involves not only a search among alternatives at a particular abstraction level for specifying choices but, in fact, a search through alternative levels of abstraction to find models of agents and actions that balance the costs and benefits of coordination appropriately.

For example, consider two robots doing deliveries, as in figure 11 (left side). Because *R*1 always delivers to the top destination and R2 to the bottom one, one strategy for coordinating is to statically assign resources (in this case, regions that contain the doors are most important). This strategy leads to figure 11 (right side), where *R*2 is always running around the long way. This organizational solution avoids any need for further coordination, but it can be inefficient, especially when *R*1 is not using its door because *R*2 is still taking the long route.

For a particular delivery, *R*1 and *R*2 might consider their time-space needs and identify that pushing their activities apart in space or time would suffice (figure 12, left side). With temporal resolution, *R*2 waits until *R*1 is done before beginning to move to and through the central door, or the robots could use information from this more abstract level to focus communication on exchanging more detailed information about the trouble spots. They could resolve the potential conflict at an intermediate level of abstraction; temporal resolution has *R*2 begin once *R*1 has cleared the door (figure 12, middle column bottom), or the robots could communicate more details (figure 12, right side), where now *R*2 moves at the same time as *R*1 and stops just before the door

to let *R*1 pass through first. Clearly, this last instance of coordination is crispest, but it is also the most expensive to arrive at and is the least tolerant of failure because the robots have less distance between them in general, so less room to avoid collisions if they deviate from planned paths.

This example illustrates that coordination can go on at different abstraction levels and that which level is correct can be very situation dependent. Thus, it is important to develop coordination techniques that can find the right level of detail. Moreover, in terms of the framework laid out in this article—of thinking about strategies for limiting the knowledge being considered during coordination—the ability to represent situations abstractly is another way of reducing the number of choices (and choice combinations) being considered (lowers *c* in our earlier formulations). Protocols that allow the incremental elaboration of choices, moreover, can be based on such a representation as another means for selectively exploring (and ignoring) options of agents (Durfee and Montgomery 1991).

Of course, there are even more strategies for coordination, even in a simple domain such as the robot-delivery task. One interesting strategy is for the robots to move up a level to see their tasks as part of a single, team task. By doing so, they can recognize alternative decompositions. For example, rather than decompose by items to deliver, they could decompose by spatial areas, leading to a solution where one robot picks up items at the source locations and drops them off at the doorway, and the other picks up at the doorway and delivers to the final destination. By seeing themselves as part of one team, the agents can coordinate to their mutual benefit (they can cooperate).

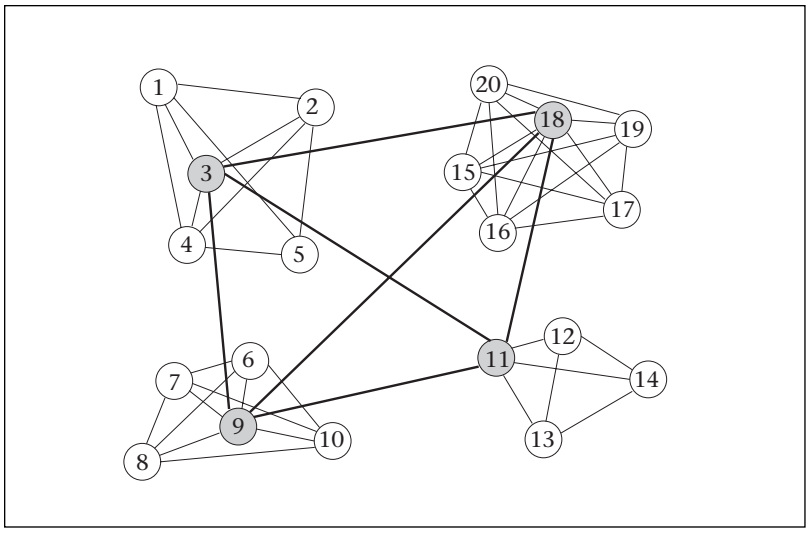

Figure 13. Example Team Hierarchy.

Nodes are clustered into four teams, and one member of each team acts as a leader and coordinates with the leaders of the other teams.

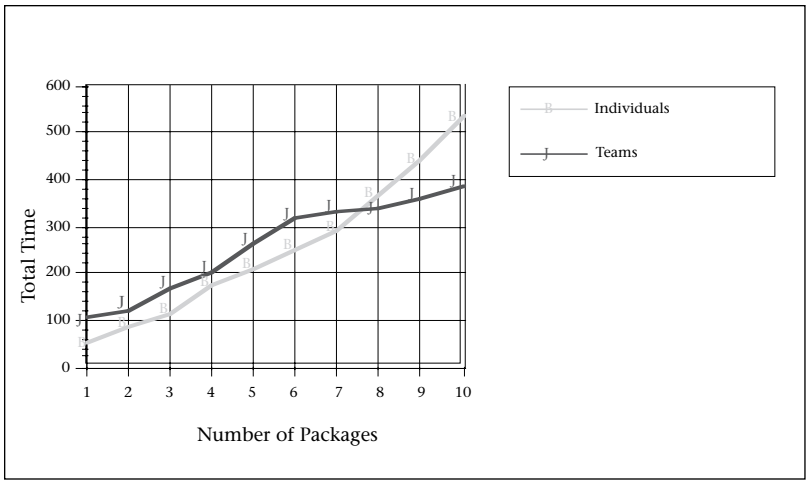

Figure 14. Experimental Results on Team Deliveries.

The total time to complete deliveries is plotted as the number of packages to be delivered (where their start and destination locations are randomly generated) grows. With a small number of deliveries, it is faster for agents to coordinate as individuals, because the number of potential conflicts (collisions) is small, than to incur the overhead of working through a team hierarchy. However, as the number of deliveries rises, it eventually becomes more cost effective to decouple much of the search by using teams despite the added overhead.

Using Teams to Simplify Coordination

We turn to one final strategy for keeping coordination practical that builds on both the ideas of using hierarchical abstractions and of ignoring agents or treating multiple agents as a single agent. Abstraction is a powerful tool for reducing complexity; for tasks that admit to abstraction such that subtasks can be elaborated independently, hierarchical distributed problem solving can solve an exponential problem in logarithmic time. That is, some problems admit to hierarchical decomposition, such as the Tower of Hanoi problem, where solving the problem of moving a stack of smaller disks off the disk that needs to be moved can be solved separately from the problem of restacking smaller disks on top of the moved disk (Knoblock 1991; Korf 1987). Multiagent problem solving can construct a plan in logarithmic time as long as the number of agents to solve the problem can grow with increasingly large problem sizes (Montgomery and Durfee 1993).

Moreover, even when strong assumptions of

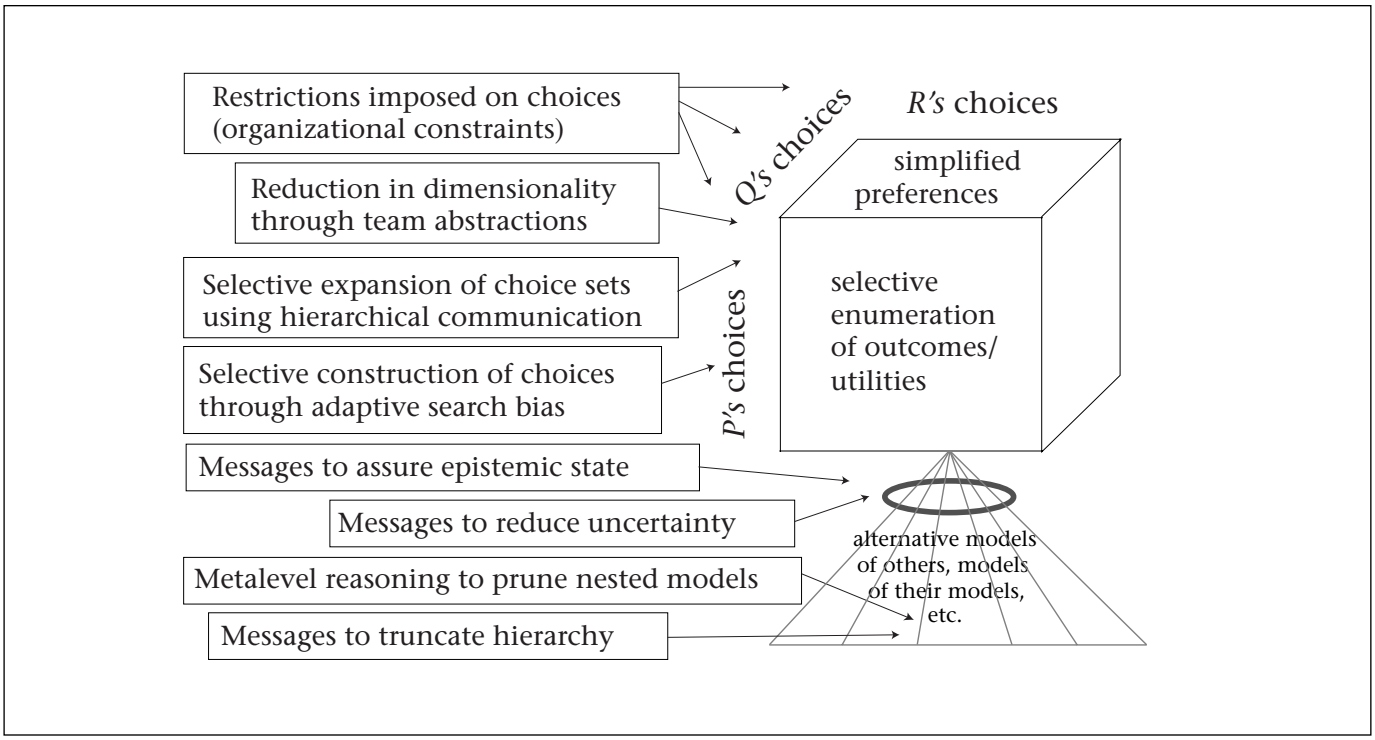

Figure 15. Strategies for Practical Coordination.

Here is a summary of the techniques described for making coordination more practical, indicating what parts of the coordination problem they affect.

subtask independence do not hold, the use of abstraction can be beneficial. For example, in coordinating 20 delivery robots, having each communicate and coordinate with all the others directly can lead to paralysis because each is overwhelmed with information. An alternative strategy is to have the agents break into teams, such that team leaders coordinate to divide (space and time) resources among the teams, and team members divide their allotment among themselves (figure 13).

Because team members must summarize their requests for team leaders, and then team leaders must pass revised team constraints back to the members, the property of subtask independence does not hold. However, despite this violation of the subtask independence assumption, as illustrated for a particular case in figure 14, the use of team-level abstraction can allow coordination costs to grow more slowly because the task becomes harder than if the individuals had to coordinate with every other agent (Montgomery and Durfee 1993).

Summary and Future Work

To summarize, what we have seen is that considering all the knowledge that an agent might have, completely thoughtful coordination might be impractical for most applications. Making coordination practical, therefore, means finding ways to not use some of the possible knowledge—to either be, or pretend to be, blissfully ignorant about some aspects of the multiagent situation (figure 15). I would claim, in fact, that the bulk of coordination research has been in developing techniques to do exactly that. The large number of techniques out there seems to me to be a reflection of the number of ways that people have found to ignore, simplify, or implicitly design away aspects of agent models to make coordination tractable. Different application domains have tended to be sensitive to ignorance of different things; hence, in general, coordination techniques appear to be tied to application domains.

My hope is that this application specificity is really not the case but that the coordination techniques are tied instead to what is safe to ignore in different domains. By characterizing coordination techniques in terms of how they reckon with the potential intractability of the coordination task, as I have done here with a small (shamelessly biased) subset of techniques, I hope to encourage further examination of previous and ongoing work (of which there is too much to comprehensively list) to

understand it not in terms of how techniques match a particular application domain but, rather, how they fit a class of domains that admit to—or even thrive on—certain kinds of ignorance that allow coordination to be practical.

Further Reading

Computational approaches to coordination among AI systems have been a focus of work in the distributed AI and multiagent system community for many years. Numerous collections of research results in the field exist, ranging from classic papers (Bond and Gasser 1988) to the papers from the International Conferences on Multiagent Systems (Demazeau 1998; Takoro 1996; Lesser 1995) and the Autonomous Agents Conferences (Sycara and Wooldridge 1998; Johnson 1997). There are in-depth treatments on the design of mechanisms that lead to multiagent systems that exhibit desirable properties (for example, Rosenschein and Zlotkin 1994), on computational organization theory (for example, Prietula, Carley, and Gasser [1998]) and on theories of nested agent knowledge (Fagin et al. 1995). Recently, the field has matured to the point where textbook-level treatments have begun to appear (Weiss 1998; O'Hare and Jennings 1996), and a journal entitled *Autonomous Agents and Multiagent Systems* is now being published.

Acknowledgments

My thanks to my current and former students, whose investigations have contributed important pieces to the overall puzzle still taking shape in this article. Their names are sprinkled liberally in the references that follow. Piotr Gmytrasiewicz and Sandip Sen specifically gave helpful suggestions about this article. This article is a revised and extended version of an invited paper presented at the 1995 International Conference on Multiagent Systems (ICMAS-95). I would like to thank my colleagues in Israel—Jeff Rosenschein, Sarit Kraus, and Moshe Tennenholtz—for feedback on the early formulation of this article and members of the multiagent system community at large since ICMAS-95 for their thoughts. This work was supported in part by the National Science Foundation (NSF) under PYI award IRI-9158473, the NSF/DARPA/NASA Digital Library Initiative CERA IRI-9411287, the Defense Advanced Research Projects Agency under contract DAAE-07-92-C-R012, and a contract with Orincon Corp.

Notes

1. This is not unlike how seemingly oblivious drivers are given the right of way on the highway by more aware and defensive drivers.

2. Recall, though, that although knowing more is better, it is not necessarily better to be known as knowing more. That is, it might be advantageous to be seen by others as being ignorant. There can be power in using knowledge that others do not know that you have! It is unlikely that the reverse (of not using knowledge that others know you have) will be a good idea.

3. Of course, if the agents truly do have infinitely deep knowledge, then this is not a simplifying assumption but, rather, a model of their true knowledge state. I address shortly the question of how such a knowledge state might come about.

4. That is, by communicating, the agents can be assured of taking complementary actions (one *A*, the other *B*), so one is sure of opening the right door (payoff of 2), and the other will get a payoff of 0. If each is equally likely to be in each of these circumstances, each has an expected (average) payoff of 1.

5. Note that prohibitions across the entire population equate to "conventions" or "social laws" (Shoham and Tennenholtz 1995) that correspond to a specialized form of organization structure.

References

Aumann, R., and Brandenberger, A. 1995. Epistemic Conditions for Nash Equilibrium. *Econometrica* 63(5): 1161–1180.

Bond, A. H., and Gasser, L., eds. 1988. *Readings in Distributed Artificial Intelligence*. San Francisco, Calif.: Morgan Kaufmann.

Cohen, P. R., and Levesque, H. J. 1995. Communicative Actions for Artificial Agents. In *Proceedings of the First International Conference on Multi-Agent Systems,* 65–72. Menlo Park, Calif.: AAAI Press.

Conry, S. E.; Kuwabara, K.; Lesser, V. R.; and Meyer, R. A. 1991. Multistage Negotiation for Distributed Constraint Satisfaction. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-21(6): 1462–1477.

Corkill, D. D. 1983. A Framework for Organizational Self-Design in Distributed Problem-Solving Networks. Ph.D. thesis, Computer and Information Science Department, University of Massachusetts.

Corkill, D. D. 1979. Hierarchical Planning in a Distributed Environment. In Proceedings of the Sixth International Joint Conference on Artificial Intelligence, 168–175. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Demazeau, Y., ed. 1998. Proceedings of the Third International Conference on Multi-Agent Systems. Washington, D.C.: IEEE Computer Society.

Durfee, E. H. 1988. *Coordination of Distributed Problem Solvers*. Boston: Kluwer Academic.

Durfee, E. H., and Lesser, V. R. 1991. Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation. *IEEE Transactions on Systems, Man, and Cybernetics* (Special Issue on Distributed Sensor Networks) 21(5): 1167–1183.

Durfee, E. H., and Lesser, V. R. 1988. Predictability versus Responsiveness: Coordinating Problem Solvers in Dynamic Domains. In Proceedings of the Seventh National Conference on Artificial Intelligence, 66–71. Menlo Park, Calif.: American Association for Artificial Intelligence.

Durfee, E. H., and Montgomery, T. A. 1991. Coordination as Distributed Search in a Hierarchical Behavior Space. *IEEE Transactions on Systems, Man, and Cybernetics* 21(6): 1363–1378.

Durfee, E. H., and Rosenschein, J. S. 1994. Distributed Problem Solving and Multi-Agent Systems: Comparisons and Examples. Paper presented at the Thirteenth International Distributed Artificial Intelligence Workshop, 28–30 July, Seattle, Washington.

Durfee, E. H., and So, Y.-P. 1997. The Effects of Runtime Coordination Strategies within Static Organizations. In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI97), 612–618. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Ephrati, E., and Rosenschein, J. S. 1994. Divide and Conquer in Multi-Agent Planning. In Proceedings of the Twelfth National Conference on Artificial Intelligence, 375–380. Menlo Park, Calif.: American Association for Artificial Intelligence.

Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. Y. 1995. *Reasoning about Knowledge*. Cambridge, Mass.: The MIT Press.

Georgeff, M. 1983. Communication and Interaction in Multi-Agent Planning. In Proceedings of the Third National Conference on Artificial Intelligence, 125–129. Menlo Park, Calif.: American Association for Artificial Intelligence.

Gmytrasiewicz, P. J., and Durfee, E. H.

1995. A Rigorous, Operational Formalization of Recursive Modeling. In *Proceedings of the First International Conference on Multi-Agent Systems,* 125–132. Menlo Park, Calif.: AAAI Press.

Gmytrasiewicz, P. J., and Durfee, E. H. 1993. Toward a Theory of Honesty and Trust among Communicating Autonomous Agents. *Group Decision and Negotiation* 2:237–258.

Gmytrasiewicz, P. J.; Durfee, E. H.; and Wehe, D. K. 1991. The Utility of Communication in Coordinating Intelligent Agents. In Proceedings of the Ninth National Conference on Artificial Intelligence, 166–172. Menlo Park, Calif.: American Association for Artificial Intelligence.

Huber, M., and Durfee, E. H. 1995. Deciding When to Commit to Action during Observation-Based Coordination. In *Proceedings of the First International Conference on Multi-Agent Systems,* 163–170. Menlo Park: AAAI Press.

Huber, M., and Durfee, E. H. 1996. An Initial Assessment of Plan-Recognition–Based Coordination for Multi-Agent Teams. In *Proceedings of the Second International Conference on Multi-Agent Systems,* 126–133. Menlo Park, Calif.: AAAI Press.

Johnson, W. L., ed. 1997. Proceedings of the First International Conference on Autonomous Agents. New York: Association of Computing Machinery.

Knoblock, C. A. 1991. Search Reduction in Hierarchical Problem Solving. In Proceedings of the Ninth National Conference on Artificial Intelligence, 686–691. Menlo Park, Calif.: American Association for Artificial Intelligence.

Korf, R. E. 1987. Planning as Search: A Qualitative Approach. *Artificial Intelligence* 33(1): 65–88.

Lee, J., and Durfee, E. H. 1995. A Microeconomic Approach to Intelligent Resource Sharing in Multiagent Systems. In *Proceedings of the First International Conference on Multi-Agent Systems,* 457. Menlo Park, Calif.: AAAI Press.

Lesser, V. R., ed. 1995. *Proceedings of the First International Conference on Multi-Agent Systems*. Menlo Park, Calif.: AAAI Press.

Lesser, V. R., and Corkill, D. D. 1981. Functionally Accurate, Cooperative Distributed Systems. *IEEE Transactions on System, Man, and Cybernetics* SMC-11(1): 81–96.

Mayfield, J.; Labrou, Y.; and Finin, T. 1996. Evaluation of KQML as an Agent Communication Language. In *Intelligent Agents, Volume 2,* eds. M. Wooldridge, J. Muller, and M. Tambe. New York: Springer-Verlag.

Montgomery, T. A., and Durfee, E. H. 1993. Search Reduction in Hierarchical Distributed Problem Solving. *Group Decision and Negotiation* 2:301–317.

O'Hare, G. M. P., and Jennings, N. R., eds. 1996. *Foundations of Distributed Artificial Intelligence*. New York: Wiley.

Prietula, M. J.; Carley, K. M.; and Gasser, L., eds. 1998. *Simulating Organizations*. Menlo Park, Calif.: AAAI Press.

Rosenschein, J. S., and Zlotkin, G. 1994. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. Cambridge, Mass.: The MIT Press.

Russell, S., and Wefald, E. 1991. *Do The Right Thing*. Cambridge, Mass.: The MIT Press.

Sen, S., and Durfee, E. H. 1995. Unsupervised Surrogate Agents and Search Bias Change in Flexible Distributed Scheduling. In *Proceedings of the First International Conference on Multi-Agent Systems*, 336–343. Menlo Park, Calif.: AAAI Press.

Shoham, Y., and Tennenholtz, M. 1995. On Social Laws for Artificial Agent Societies: Off-Line Design. *Artificial Intelligence* 73(1): 231–252.

Smith, R. G. 1980. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers* C-29(12): 1104–1113.

So, Y.-P., and Durfee, E. H. 1996. Designing Tree-Structured Organizations for Computational Agents. *Computational and Mathematical Organization Theory* 2(3): 219–246.

Sycara, K. 1989. Multiagent Compromise via Negotiation. In *Distributed Artificial Intelligence, Volume 2*, eds. L. Gasser and M. Huhns, 119–137. London: Pittman.

Sycara, K. P., and Wooldridge, M., eds. 1998. Proceedings of the Second International Conference on Autonomous Agents. New York: Association of Computing Machinery.

Sycara, K.; Roth, S.; Sadeh, N.; and Fox, M. 1991. Distributed Constrained Heuristic Search. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-21(6): 1446–1461.

Takoro, M., ed. 1996. *Proceedings of the Second International Conference on Multi-Agent Systems*. Menlo Park, Calif.: AAAI Press.

Vidal, J. M., and Durfee, E. H. 1998. The Moving Target Function Problem in Multi-Agent Learning. In *Proceedings of the Third International Conference on Multi-Agent Systems*, 317–324. Washington, D.C.: IEEE Computer Society.

Vidal, J. M., and Durfee, E. H. 1996. Using Recursive Agent Models Effectively. In *Intelligent Agents*, Volume 2, eds. M. Wooldridge, J. Muller, and M. Tambe, 171–186. New York: Springer-Verlag.

Vidal, J. M., and Durfee, E. H. 1995. Recursive Agent Modeling Using Limited Rationality. In *Proceedings of the First International Conference on Multi-Agent Systems*, 376–383. Menlo Park, Calif.: AAAI Press.

Weiss, G., ed. 1998. *Multiagent Systems: A Modern Approach to DAI*. Cambridge, Mass.: MIT Press.

Weiss, G., and Sen, S., eds. 1996. *Adaptation and Learning in Multi-Agent Systems*. Lecture Notes in AI 1042. New York: Springer-Verlag.

Wellman, M. P. 1993. A Market-Oriented Programming Environment and Its Application to Distributed Multicommodity Flow Problems. *Journal of Artificial Intelligence Research* 1:1–23.

Yokoo, M.; Durfee, E. H.; Ishida, T.; and Kuwabara, K. 1992. Distributed Constraint Satisfaction for Formalizing Distributed Problem Solving. In Proceedings of the Twelfth International Conference on Distributed Computing Systems, 614–621. Washington, D.C.: IEEE Computer Society.

Edmund Durfee is an associate professor of electrical engineering and computer science at the University of Michigan and holds a joint appointment at the School of Information. His research interests are in distributed AI, planning, multiagent systems, and real-time problem solving applied to problems ranging from digital libraries to cooperative robotics, from assistance technologies to electronic commerce. He received a B.A. in 1980 from Harvard University and a Ph.D. in 1987 from the University of Massachusetts at Amherst. He was named a Presidential Young Investigator in 1991. He is an associate editor for the Institute of Electrical and Electronics Engineers *Transactions on Systems, Man, and Cybernetics* and for the *International Journal on Autonomous Agents and Multiagent Systems,* served as a program cochair for the 1998 International Conference on Multiagent Systems (ICMAS), and will serve as the chair for the 2000 ICMAS. His e-mail address is durfee@umich.edu.