Blissful Ignorance: Knowing Just Enough to Coordinate Well

Edmund H. Durfee

EECS Dept., University of Michigan
Ann Arbor, MI 48109-2110
durfee@umich.edu

Abstract

To coordinate, intelligent agents might need to know something about each other, and about themselves, and about how others view themselves, and how they view others, and how they think others view others, and so on. Taken to an extreme, the amount of knowledge an agent might possess to help make sense of an interaction might outstrip the agent's limited reasoning capacity (its available time, memory, etc.). 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---that is, by keeping agents selectively ignorant. In this paper, I try to put various such efforts in perspective, characterizing different ways in which agents can beneficially be kept ignorant of some knowledge. *

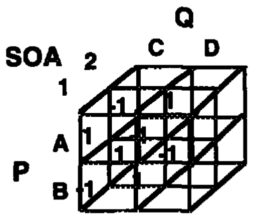
Introduction

It has been said that ignorance is bliss. Certainly, people who know too 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 is to strike a balance between designing agents that have enough knowledge to coordinate well, and no more knowledge than that, lest the knowledge over and above what is needed impede the timely decisionmaking of the agent. This paper is about characterizing techniques that allow agents to know just enough to coordinate well.

Before we begin, however, let me first be a little clearer about what I mean by "agent" and by "coordinate." In keeping with much of the current usage of the term, I will consider an agent to be an entity that is capable of acting in its environment to satisfy its goals. 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 outcomes of actions, states of the world). We can capture these three characteristics in a variety of notations, but here let me just represent them in terms of a payoff matrix, where agent P can take actions A or B, has beliefs about whether the world is in state of affairs 1 or 2, and prefers to take actions that lead to higher payoffs (the numbers inside the matrix).

		SOA 1	SOA 2
P	A	1	-1
	B	-1	1

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 those other agents when making its own choice. I will refer to this---the taking into account the choices of others when deciding what to do---as an agent *coordinating* with those others. Note that coordination does not imply cooperation, or competition for that matter. In fact, it may not even be symmetric, in that one agent can coordinate with another without the other coordinating with the first. From the perspective of agent P, for example, it might need to consider agent Q's possible choices when making its own decision, as shown here.



To make its decision, P needs to determine whether the state of affairs is 1 or 2, and whether Q is likely to do C or D, 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, like a coin that gets flipped, then this 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. Of course, if P knows what Q prefers to do in different states of affairs, then P might be better off modeling the probability of Q taking an action given Q's belief of the state of affairs, but then P needs to model what Q's beliefs in the state of affairs are, given what P thinks the state of affairs is! In turn, then, P might be better off modeling Q as following its own detailed choices, beliefs, and preferences, which could be captured, for example, as a payoff matrix. But now, if P's model of Q's payoff matrix includes Q's uncertainty about P's choices, then P

* This work was supported, in part, by the National Science Foundation under PYI award IRI-9158473, by the NSF/ARPA/NASA Digital Library Initiative CERA IRI-9411287, by ARPA under contract DAAE-07-92-C-R012, and by a contract with Orincon Corp.

has to model Q's model of P. And such nested modeling could go on arbitrarily deeply!

Coming to grips with arbitrarily nested models has been an ongoing battle in research communities interested in coordination among agents. One successful strategy has been to assume an infinite nesting of these models, and thus to seek fixpoint solutions for the choices of all of the agents involved. This has been the standard approach in game theory, for example, in identifying equilibrium solutions, a topic we will return to shortly. Practical limitations in communication often invalidate assumptions about common knowledge (which gives rise to infinitely-nested models). In such cases, alternative methods for representing and using the nested models are needed. One such method is the Recursive Modeling Method.

Recursive Modeling Method

The Recursive Modeling Method (RMM) captures relevant knowledge for coordination decisionmaking into a framework that admits to a rigorous solution method (Gmytrasiewicz and Durfee, 1995). RMM models an interaction as a payoff matrix, and so nested models create nested instances of payoff matrices. As an example (see (Gmytrasiewicz and Durfee, 1995) for a more complete description of RMM), consider the following situation (Figure 1). A robot R1 faces a choice between pursuing goal G1 or pursuing goal G2 or doing Something else. Doing G1 has a payoff of 2, but doing it costs R1 1. Doing G2 has payoff of 5, and cost of 2. Doing S has no payoff and no cost. If it were acting alone in the world, R1's choice would be obvious---to do G2. But it isn't alone; R2 is also in the world. The good news is that R1 gets the benefits of all the goals accomplished, so it can be better off thanks to R2. Trouble is, R1 does not know whether R2 is like R1 (call it a type A robot), or R2 is another type of robot (type B). It does know what R2's expected payoffs would be in either case, but it does not know anything about what R2 might expect R1 to do. This lack of knowledge is represented as an equiprobability distribution, as is depicted in Figure 2.

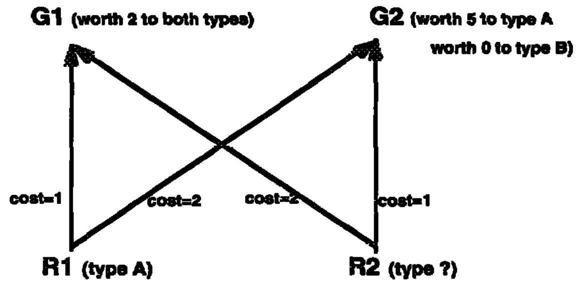

G1 (worth 2 to both types)
G2 (worth 5 to type A worth 0 to type B)
cost=1
cost=2
cost=2
cost=1
R1 (type A)
R2 (type ?)

Figure 1: An Abstract Coordination Situation

RMM works by propagating expectations from the leaves to the top. From the left leaf, R1 thinks R2 (if type B) would think R1 is equally likely to do G1 or S (something else), in which case R1 would expect R2 to prefer to do S. From the right leaf, R1 thinks R2 (if type

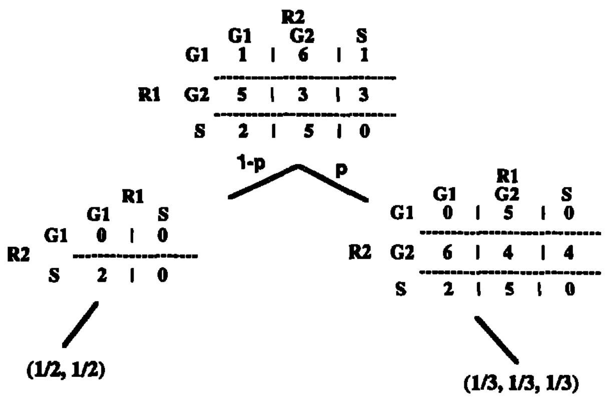


Figure 2: A Possible RMM Hierarchy

A) thinks R1 is equally likely to do G1, G2, or S, and so R1 expects R2 to take the action with the highest average payoff, which is G2. Working upward, then, if R1 thinks R2 is equally likely to be type A or B, then it expects R2 is equally likely to do G2 or S, and R1 should take action G1, with an expected payoff of 3.5.

Of course, the nesting of knowledge can be deeper, leading to branchy, tall trees which take longer to construct and solve. For the time being, assume that the trees of nested beliefs must be finite; infinite trees would require the establishment of common knowledge, which can be impractical in realistic situations (Halpern & Moses, 1984). Even finite trees can get quite large, however, as is the case if R1 considers that R2, if of type A, will think that R1 could be of type A or B, and if of type A, will think R2 could be of type A or B, and so on, as shown in Figure 3. Note that, even as deeper nesting is done, R1's choice (of doing G1 with expected utility of 3.5) remains constant.

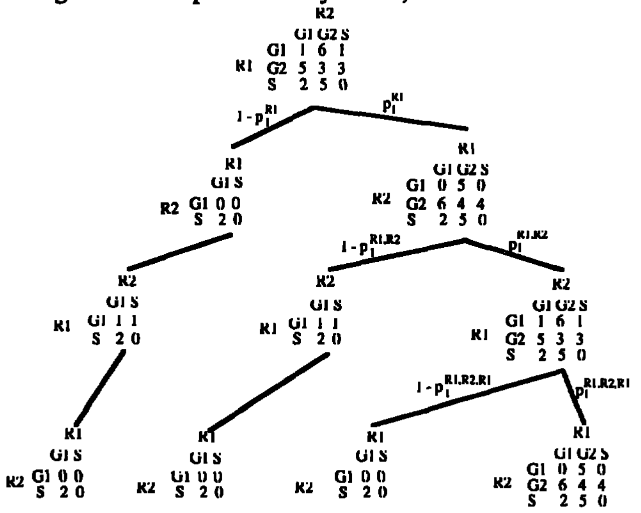


Figure 3: Deeper, Branchier Recursive Modeling

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. If we assume each of the n agents can take a actions, then there are an actions to identify. Since each action choice can correspond to a plan on the part of an agent, that means there are an plans to formulate. And for each of the a^n combinations of actions, the value (utility) of the outcome must be computed. The above 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 agent types in the running example) and l is the depth of the recursive models available to the agent.

As an agent knows more, therefore, it must do much more computation. Since 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 down the knowledge being used (Figure 4). We can be selective about the nested knowledge we use, or even obviate its use by exploiting communication. We can simplify the utility calculations, and trim down the number of options evaluated for each agent. We can even reduce the dimensionality of an interaction by ignoring agents or viewing groups of agents as individuals. 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 paper, I consider examples of such methods, working from the bottom of Figure 4 upward.

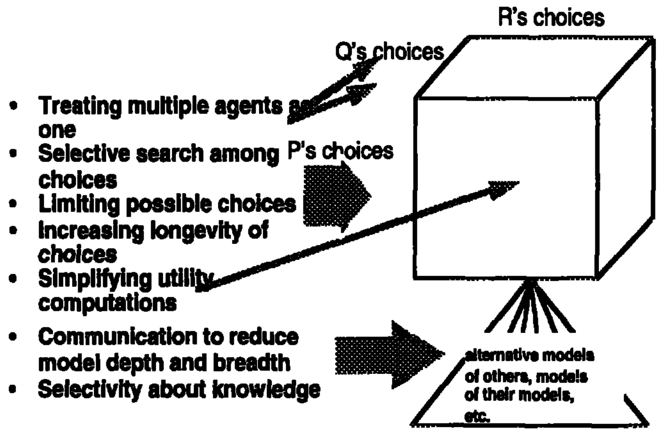

R's choices
Q's choices
• Treating multiple agents as one
• Selective search among choices
P's choices
• Limiting possible choices
• Increasing longevity of choices
• Simplifying utility computations
• Communication to reduce model depth and breadth
• Selectivity about knowledge
alternative models of others, models of their models, etc.

Figure 4: Strategies for Making Coordination Practical

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 the latter 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, undesirable states can be pruned because agents will not act to get into those states. With nested models, however, the models exist regardless of their desirability; ignoring the fact that another agent might be plotting against you will not make it go away. 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, rather than pruning undesirable states *per se*.

Our preliminary results employing such a strategy indicate that this approach holds promise, even for simple application domains. Vidal & Durfee (1995) show that, even in a simple pursuit task, the models of interactions can be quite large. By using heuristic estimates (based on previous experience) about the impact of expanding different parts of the nested models, we can achieve a high level of coordination quality using only a small subset of the nested models.

Exploiting Communication

Whereas using meta-level knowledge to be selective about which models to use is one strategy for cutting short the use of nested models, another more common method is to obviate the need for nested models by having agents explicitly tell each other about their intentions (or at least about constraints on what they will be doing). Such an approach has formed the backbone of the bulk of work in multiagent planning, for example, where agents separately form their own plans, and then communicate to identify possible conflicts or cooperative opportunities.

The benefits of such communication are clear when captured in the RMM framework. By revealing information about itself, an agent simplifies its models of others, either by reducing uncertainty about the world (and hence reducing the branching factor) or uncertainty about what others will be doing (hence reducing the requisite modeling depth). For example, if R1 considers telling R2 "I will pursue G2," then R1 would model the resultant mental situation as being truncated, since it knows exactly that R2 will consequently expect R1 to pursue G2 (or S, if R2 does not understand what G2 is). Using the RMM solution methods on the resultant model (Figure 5), R1 can conclude that R2 will be equally likely to pursue G1 or S (both choices yield equal expected payoffs for either case of R2 being type A or B), so R1 indeed should pursue G2 with an expected payoff of 4. Not only did the message allow R1 to pursue a different goal, but R1 now has a higher expected payoff thanks to the communication. Naturally, this assumes that the message is delivered and believed (Gmytrasiewicz & Durfee, 1993).

Also, note that R1 could similarly have triggered this improvement if it could expect R2 to observe and infer

R1's choice of goal based on R1's initial actions. By using plan recognition, agents can similarly prune the space of alternative models under consideration, albeit taking into account the uncertainty of such inferences (Huber & Durfee, 1995).

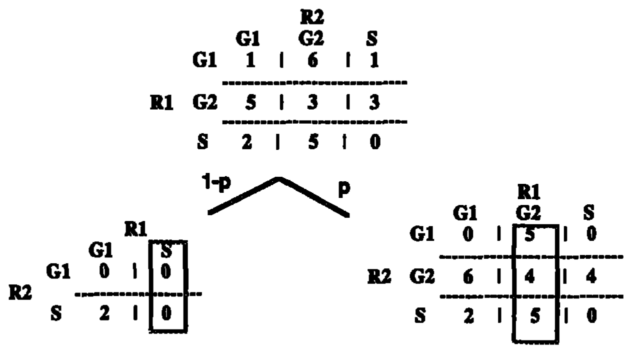


Figure 5: Truncated Models Due to Communication

Epistemic States for Well-Defined Coordination

In the previous section, 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 & 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 independently 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. While their analyses get rather complicated, 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, R1 recognized that by telling R2 about R1's intention to pursue G2, R1 could increase its own expected payoff. Moreover, in this case, R1 got this higher payoff by doing what it had told R2 it would do (it did not need to lie). If R1 is correct in its knowledge of R2's payoff matrices, of R2's rationality, and of R2's correctly receiving R1's message, then R1 in fact knows that the agents are in a Nash equilibrium if it has itself been rational in choosing its action and truthful in revealing it (this is pretty self-evident, although we are constructing a more formal proof in the spirit of Aumann and Brandenberger). To see that R1's doing G2, and R2 being indifferent between G1 and S (as anticipated by R1) is indeed a Nash equilibrium, consider the matrices containing both agents' payoffs in Figure 6, for each type that R2 could be. In both cases, neither agent has an incentive to change its strategy assuming the other persists in its strategy, so the strategy combination is a Nash equilibrium.

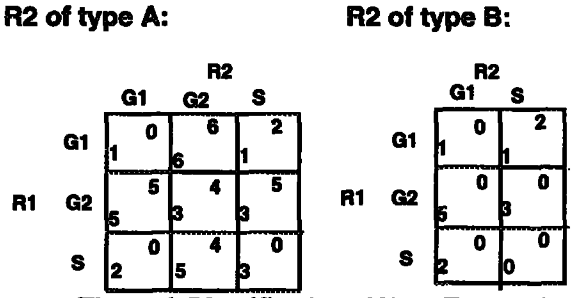


Figure 6: Identification of Nash Equilibria

Notice, of course, that there are other possible Nash equilibria. For example, if R2 is of type A, then R1 pursuing G1 and R2 pursuing G2 is also a Nash equilibrium, and moreover is Pareto efficient (both agents are better off with that equilibrium). Note, however, that, if R2 is type B, R1 should not choose G1; G2 strictly dominates G1 and S in fact. Thus, we would expect that someone who wanted this multiagent system to find a good (or optimal) Nash equilibrium might impose a protocol on the agents to not only truthfully reveal intentions, but also to remove ambiguity by having agents truthfully inform each other about their types. In fact, by performing such a knowledge-level analysis, coupled with well-defined criteria for properties of the coordination "solution" expected of the agents, a system designer can craft a protocol that can ensure desired multiagent behavior.

Preference Simplification and Selective Search

At this point, let me turn away from strategies for reducing the nested, branching models of others' models of others'... to focus on strategies for keeping the representation of an interaction (or, in RMM terms, keeping each payoff matrix) tractable. Recall that, in the worst case, every possible combination of the *a* actions for the *n* agents must be evaluated, meaning a^n evaluations. Fortunately, this number can be reduced if portions of the combination space can be safely ignored.

One simple way of avoiding the exhaustive enumeration is to simplify the evaluation to return only two possible values, corresponding to *good* and *no good*. In effect, this reduces the search for an optimal choice of action given others' actions into a search for a satisficing choice; the optimization problem becomes a satisfaction problem. Much work in multiagent systems has benefited from this kind of simplification, taking advantage of algorithms for constraint satisfaction to solve, for example, problems in distributed resource allocation and scheduling.

As a result of this kind of simplification, the search among agents for a coordinated solution to their problem amounts to a search through their options until a satisfactory one is found. 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.

For example, in the application domain of distributed meeting scheduling, there could be several strategies for how agents go about searching through the space of possible meeting times to propose to each other (Sen & Durfee, 1995). Two possible strategies are to simply move through the available times chronologically and schedule a meeting as early as possible, or to 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 calendars that look different, the latter 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. Thus, not only does an agent's choice of strategy for searching the options impact the quality and cost of its schedule, but the agent also must be capable of adapting its strategy as circumstances change.

Selectively searching among the space of joint options is most traditionally associated with the term "negotiation," implying the iterative exchange of proposals among agents to converge on a satisfactory or (locally) optimal collective decision. This has been, and will continue to be, a very fertile research area in the field, in no small way due to all of the possible strategies for making negotiation practical depending on the needs of an application, ranging from simplifying the preferences and adapting the search strategy, to simplifying the proposals (for example, using prices to summarize agents' current/future plans (Lee & Durfee, 1995)), to using heuristics to generate new proposals based on feedback about prior proposals, 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 like 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 paper) by focusing on how each kind attempts to simplify the coordination problem.

Constraining Choices

Rather than searching through the space of (collective) choices incrementally, which can be very time-consuming and communication-intensive, agents can instead establish (at the outset, or at intervals) constraints on their choices so as to simplify how they should model themselves and others. This idea 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. A description of an organization captures the preferences and responsibilities of its members, providing strong clues to the choices of actions the members might make. The organization might also impose prohibitions on certain kinds of actions, both for particular niches in the organization and for all members of the organization.[1] Among the challenges in designing an organization is determining how to decompose tasks into separate roles, so as 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 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 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 & Durfee, 1993):

$$T(N,k,\gamma,m)=\left\{\begin{matrix}(k+1)l\delta+(l+m)\tau\longrightarrow\gamma\le 1\\ \\ 2l\delta+(kl+m)\tau\longrightarrow\gamma>1\end{matrix}\right\}$$

For example, for $N=32$, $m=2$, and any γ, $k=4$ outperforms $k=2$ and $k=16$. But this 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. But redundancy opens the door to possible inefficiencies, as agents duplicate each others' work. Duplication can be avoided if agents are able to coordinate dynamically, but this in turn incurs overhead and assumes sophisticated agents. So 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 child in a tree-like organization as an assignment that tolerates the failure of *o* children. For different levels of o-redundancy, along with the previously described parameters, we can measure the organization performance variance (OPV) as the difference between the performance time of the organization if agents coordinate optimally and if agents coordinate as poorly as possible. Analyses of the performance of two organizations, for example, are summarized in Figures 7 and 8. Note how, in the first case, OPV decreases with increasing failure rate, while it doe not do so in the second case, indicating that sophisticated coordination abilities are needed over a wider range of situations in the second case. 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 as agent failure rate increases.

[1] Note that prohibitions across the entire population equate to "conventions" or "social laws," which correspond to a specialized form of organization structure.

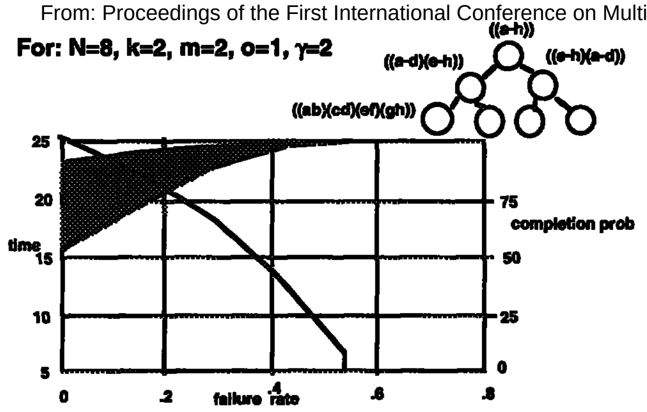


Figure 7: Binary Tree Organization

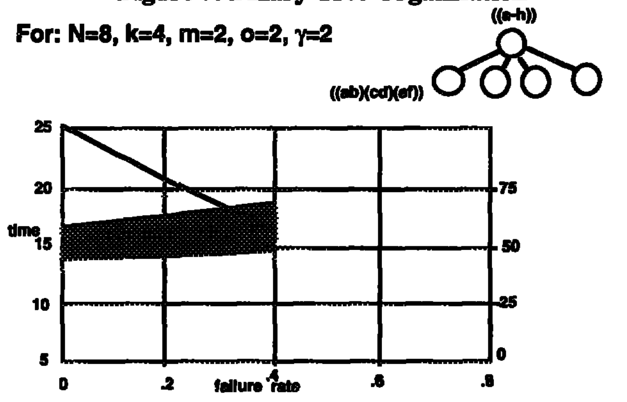


Figure 8: Flat Tree Organization

These results and others (So & Durfee, 1995) illustrate how the design of an organization should be predicated on the available sophistication of the agents that might populate it, as well as the expected task-environment of the agents. 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.

Hierarchical Elaboration of Choices

So far, 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 approaches where agents work 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 at the organization level). Generally speaking, agents can make commitments at any of a number 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 9. The choice of what kinds of commitment(s) agents 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.

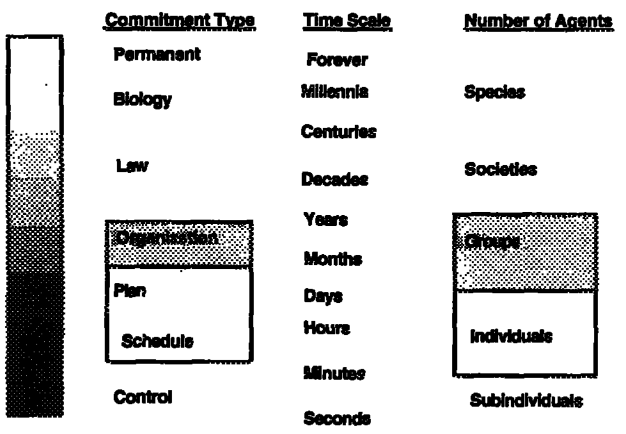


Figure 9: Commitment Spectrum

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 level of abstraction, but in fact a search through alternative levels of abstraction to find models that balance the costs and benefits of coordination appropriately.

For example, consider 2 robots doing deliveries as in Figure 10 (left side). Since R1 always delivers to the top destination, and R2 to the bottom one, one strategy for coordinating is to statically assign resources (in this case, the doors are most important). This leads to Figure 10 (right side), where R2 is always running around the long way. This solution avoids any need for further coordination, but it can be inefficient, especially when R1 is not using its door, since R2 is still taking the long route.

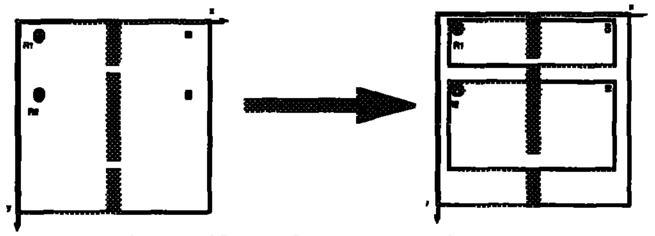


Figure 10: An Organization Solution

For a particular delivery, R1 and R2 might consider their time/space needs, and identify that pushing their activities apart in space or time would suffice (Figure 11, left side). With temporal resolution, R2 waits until R1 is done before beginning to move 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 R2 begin once R1 has cleared the door (Figure 11, middle column bottom). Or they could communicate more details (Figure 11, right side), where now R2 moves at the same time as R1, and stops just before the door to let R1 pass through first. Clearly, this last instance of coordination is crispest, but it is also the most expensive to arrive at and the least tolerant of failure, since the robots have less distance between them in general, so less room to avoid collisions if they deviate from planned paths.

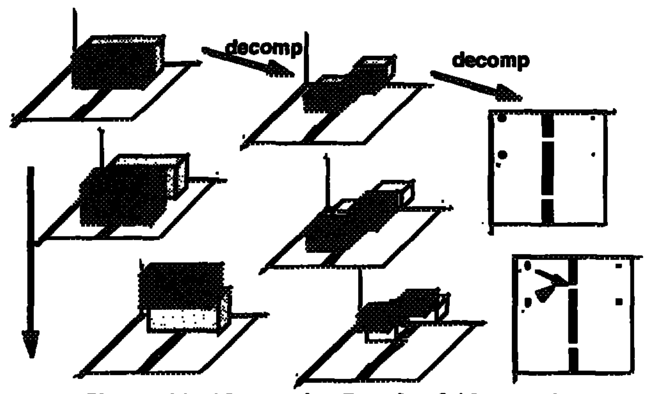


Figure 11: Alternative Levels of Abstraction

This example illustrates that coordination can go on at different abstraction levels, and that which level is right 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 paper---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 *a* 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 & Montgomery, 1991).

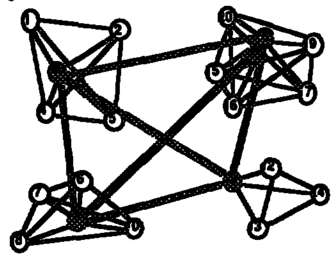
Of course, there are even more strategies for coordination even in a simple domain such as the robot delivery task. One interesting strategy is for the robots to move up a level---to see their tasks as part of a single, team task. By doing so, they can recognize alternative decompositions. For example, 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.

Using Teams to Simplify Coordination

We turn to one final strategy for keeping coordination practical---reducing the dimensionality of the interactions (in RMM terms, of the payoff matrices) by ignoring agents or treating multiple agents as a single agent. To keep things brief, we will skip strategies for deciding which agents to ignore; such strategies can follow from notions such as dominance analysis in game theory, where an agent's choice is clear regardless of the choices of one or more others. Instead, we concentrate on the notion of using team abstractions.

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 (as long as the number of agents to solve the problem can grow with increasingly large problem sizes) (Montgomery & Durfee, 1993).

Even when strong assumptions of subtask independence do not hold, moreover, the use of abstraction can be beneficial. For example, in coordinating 20 delivery robots, having each communicate and coordinate with all of the others directly leads to paralysis as 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.



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. Yet, despite this, as seen in Figure 12, the use of team-level abstraction allows coordination costs to grow more slowly as the task becomes harder than if the individuals had to coordinate with every other agent (Montgomery & Durfee, 1993).

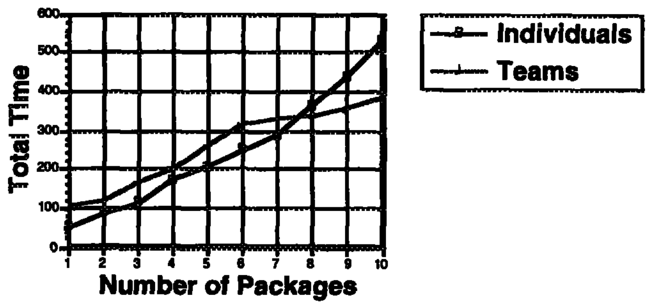


Figure 12: Experimental Results on Team Deliveries

Summary and Future Work

To summarize, what we have seen is that, considering all of 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 13). 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 work. 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 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.

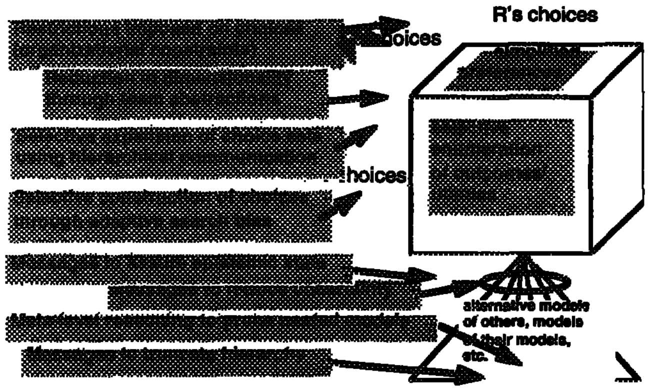


Figure 13: Strategies for Practical Coordination

Acknowledgments

My thanks to my current and former students, whose investigations have contributed important pieces to the overall puzzle still taking shape in this paper. Also thanks my colleagues in Israel---Jeff Rosenschein, Sarit Kraus, and Moshe Tennenholtz---for feedback on earlier versions.

References

Aumann, R. and Brandenberger, A. 1995. Epistemic Conditions for Nash Equilibrium. *Econometrica* (to appear).

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

Durfee, E. H., and Rosenschein, J.S. 1994. Distributed Problem Solving and Multi-Agent Systems: Comparisons and Examples. Proceedings of the Thirteenth International Distributed Artificial Intelligence Workshop, pages 94-104, July 1994.

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., and Durfee, E. H. 1995. A Rigorous, Operational Formalization of Recursive Modeling. To appear in Proceedings of the First International Conference on Multi-Agent Systems, San Francisco, CA. AAAI Press.

Halpern, J.Y., and Moses, Y., 1984. Knowledge and Common Knowledge in a Distributed Environment. In Proceedings of the Third ACM Conference on Principles of Distributed computing.

Huber, M., and Durfee, E. H. 1995. Deciding When to Commit to Action During Observation-based Coordination. To appear in Proceedings of the First International Conference on Multi-Agent Systems, San Francisco, CA. AAAI Press.

Lee, J., and Durfee, E. H. 1995. A Microeconomic Approach to Intelligent Resource Sharing in Multiagent Systems. To appear in Proceedings of the First International Conference on Multi-Agent Systems, San Francisco, CA. AAAI Press.

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

Sen, S., and Durfee, E. H. 1995. Unsupervised Surrogate Agents and Search Bias Change in Flexible Distributed Scheduling. To appear in Proceedings of the First International Conference on Multi-Agent Systems, San Francisco, CA. AAAI Press.

So, Y-P., and Durfee, E.H. 1995. Organization Design and Local Sophistication. Forthcoming.

So, Y-P., and Durfee, E.H. 1994. Modeling and Designing Computational Organizations. Working Notes of the AAAI Spring Symposium on Computational Organization Design, March 1994.

Vidal, J. M., and Durfee, E. H. 1995. Recursive Agent Modeling Using Limited Rationality. To appear in Proceedings of the First International Conference on Multi-Agent Systems, San Francisco, CA. AAAI Press.