Emergent Coordination through the Use of Cooperative State-Changing Rules

Claudia V. Goldman and **Jeffrey S. Rosenschein***
Computer Science Department
Hebrew University
Givat Ram, Jerusalem, Israel
clag@cs.huji.ac.il, jeff@cs.huji.ac.il

Abstract

Researchers in Distributed Artificial Intelligence have suggested that it would be worthwhile to isolate "aspects of cooperative behavior," general rules that cause agents to act in ways conducive to cooperation. One kind of cooperative behavior is when agents independently alter the environment to make it easier for everyone to function effectively. Cooperative behavior of this kind might be to put away a hammer that one finds lying on the floor, knowing that another agent will be able to find it more easily later on.

We examine the effect a specific "cooperation rule" has on agents in the multi-agent Tileworld domain. Agents are encouraged to increase tiles' degrees of freedom, even when the tile is not involved in an agent's own primary plan. The amount of extra work an agent is willing to do is captured in the agent's *cooperation level*. Results from simulations are presented. We present a way of characterizing domains as multi-agent deterministic finite automata, and characterizing cooperative rules as transformations of these automata. We also discuss general characteristics of cooperative state-changing rules. It is shown that a relatively simple, easily calculated rule can sometimes improve global system performance in the Tileworld. Coordination emerges from agents who use this rule of cooperation, without any explicit coordination or negotiation.

Introduction

Distributed Artificial Intelligence (DAI) is concerned with effective agent interactions, and the mechanisms by which these interactions can be achieved. Researchers in DAI have taken many approaches to this overall question, considering in particular explicit coordination and negotiation techniques (Smith 1978; Malone, Fikes, & Howard 1988; Kuwabara & Lesser 1989; Conry, Meyer, & Lesser 1988; Kreifelts & Martial 1990; Durfee 1988; Sycara 1988; 1989; Kraus & Wilkenfeld 1991; Zlotkin & Rosenschein 1993b; Ephrati & Rosenschein 1993), as well as implicit modeling of other agents' beliefs and desires (Genesereth, Ginsberg, & Rosenschein 1986; Gmytrasiewicz & Durfee 1992; Kraus 1993; Grosz & Kraus 1993).

There have also been repeated attempts by researchers to establish norms of cooperative behavior, general rules that would cause agents to act in ways conducive to cooperation. The search space for multi-agent action is large, and cooperative behavior on the part of agents would ideally act to limit this search space. These investigations into cooperative behavior have generally taken the approach of shaping agents' plans in particular directions, such that other agents could interact appropriately. An agent that acts predictably, shares its information, and defers globally constraining choices as long as possible, will be an easier one with which to coordinate. Work in this area includes early research by Davis and his colleagues at MIT (Davis 1981), and some of the RAND work on cooperative behavior in the air traffic control domain (McArthur, Steeb, & Cammarata 1982). Multi-agent reactive systems have also been analyzed, where solutions are arrived at dynamically by reactive agents (eco-agents) in multi-agent environments (Ferber & Drogoul 1992). More recently, Tennenholtz, Shoham, and Moses have considered how social laws for artificial agent societies could be developed and evaluated (Tennenholtz & Moses 1989; Shoham & Tennenholtz 1992b; 1992a).

While these streams of research have considered how agents' plans could be adapted to maximal cooperative effect, we take a different approach to the question. Instead of asking how an agent might temper its own goal-satisfying behavior to be cooperative, we ask how an agent might *transform the world* in a cooperative manner, at the same time that it is pursuing its own goal. We explore *cooperative state-changing rules*, that are imposed on the agents' behaviors as meta-rules. Those meta-rules don't have any influence on the primary actions that the agents are going to execute. Rather, they induce the agents to perform *extra work* that will transform the world into one in which the work of all agents might be done more easily.

*This research has been partially supported by the Israeli Ministry of Science and Technology (Grant 032-8284).

Tileworld Interactions

The Domain

Consider agent interactions in a multi-agent version of the Tileworld (Pollack & Ringuette 1990). Agents can move only through navigable discrete squares; moving from one square to another costs one unit. Agents are programmed to roam the grid and push tiles into holes. In our simulations, we considered two ways in which agents decide which tile to go after. In one variation, they choose the closest (Euclidean distance) tile, compute and traverse their path to it, then push it to the hole that is closest to the tile (again, Euclidean distance). The computation of distance is a (lower-bound) heuristic, since it doesn't take into account barriers, other agents, and tiles (but it is quick to compute). In the second variation, agents are assigned tiles and holes arbitrarily by the programmer, and stop when they finish their assignments. This use of the Tileworld is non-standard, in that we are *not* focusing here on the architecture (reactive or otherwise) of the agents. Instead, we are using the Tileworld as an interesting, constrained domain that helps us understand implicit cooperative behavior.

Strongly-Coupled Interactions

The goals of separate agents in the multi-agent Tileworld are highly interconnected, primarily because the constraints on movement are so severe. Agents need to laboriously go around barriers to get into position and push a tile into a hole. Consider, for example, the simple interaction shown in Figure 1 (a variation of an example from (Zlotkin & Rosenschein 1993a)).

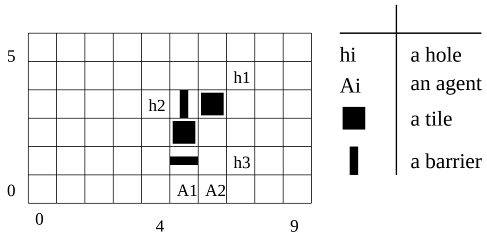

Figure 1: Strongly-Coupled Interactions

Assume that agent A_1 wants to fill holes 1 and 2, and that agent A_2 wants to fill holes 2 and 3 (perhaps the agents were assigned these goals *a priori*). For either agent to accomplish its goal, it would need to carry out a large number of movements to position the tiles and itself appropriately. For example, for A_1 to fill its holes alone, it needs to move 17 steps (assuming A_2 is not in its way). Similarly, A_2, alone in the world, would need to move 26 steps to fill its holes. However, if they work together, they can satisfy A_1 completely (and A_2 partially) by going 12 steps (or satisfy A_2 completely and A_1 partially).

The highly constrained nature of the multi-agent Tileworld provides ample opportunity for cooperative behavior, as the above example shows (e.g., one agent pushes a tile away from a barrier, while the other then pushes it perpendicularly). However, finding these multi-agent optimal plans tends to be a very difficult task. Instead, we examine the possibility that the cooperative behavior exhibited in the above example can be approximated by giving the agents an inclination towards sociable behavior. We will induce this kind of behavior through a meta-rule that causes the agents to help one another implicitly, without having to search for an optimal multi-agent plan. Ideally, when the agents are acting sociably, their combined activity will approach the optimal solution that they would have found had they carried out full multi-agent planning, but at a fraction of the computational cost.

A Rule for Sociable Behavior in the Tileworld

In general, a rule for sociable behavior can take several forms. In the simplest case, an agent may have two courses of action that he perceives as equivalent; if other agents would prefer him to carry out specifically one of those courses of action, he might do so in order to be cooperative. Sometimes, however, we may design agents to actually carry out extra work, to improve the environment for other agents (the amount of extra work subject to the designer's discretion). This latter kind of rule is the one that interests us here.

Obviously, if one designer is building all agents, as in a cooperative problem solving scenario, then such a cooperative meta-rule will have clear utility if it improves overall system performance. If separate designers, with separate goals, are building the agents, there may still be justification for their putting in such a meta-rule, under certain circumstances; however, we do not consider these issues (such as *stability* of the cooperative rule) in this paper.

In the Tileworld domain, it is better for every agent to have tiles freely movable, i.e., less constrained by barriers, so that tiles can be pushed into holes more easily. Each tile has a *degree of freedom* associated with it, the number of directions in which it can be pushed, either zero, two, or four. When we want agents to act cooperatively, we induce them to free tiles, by increasing the tiles' degree of freedom from two to four.

The key point in any given domain is to identify the state characteristics that allow goals to be achieved more easily. Cooperative agents are those that tend towards moving the world into these more conducive, less constrained configurations. In general, a world with more freed tiles is better than one that has constrained tiles. Although it may not be cheap for an agent to free a given tile, it may sometimes be possible to exert a small amount of extra effort, free a tile, and save another agent a large amount of work. In particular, if agent A_i is on its way to push a tile into a hole, and there is a constrained tile close by that can be freed, then the agent might free it.

This kind of cooperation does not require that the agents communicate, nor that they have a model of the other agents' specific plans, beliefs, or goals (though we do assume that agents know the current status of the grid). Cooperation will emerge out of the sociable behavior of the individual agents.

How much extra work should an agent be willing to do to act sociably? In our domain, how close does a constrained tile need to be for an agent to go out of its way and free it? In general, the *extra work* that the agent may do is the movement outside one of its minimal paths and back, so as to free a tile. We call this amount of extra work the *cooperation level* of the agent.

Simulations

We have run simulations using the MICE distributed agent testbed (Montgomery *et al.* 1992) to statistically analyze the efficacy of different cooperation levels (Hanks, Pollack, & Cohen 1993).

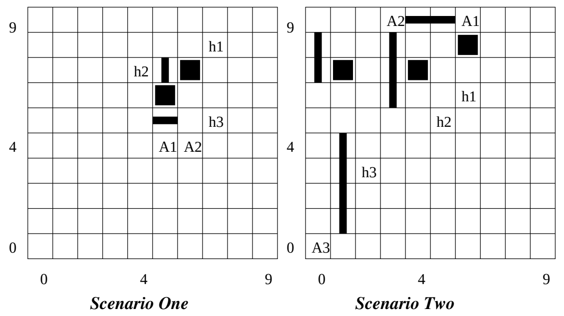

Figure 2: Simulations

In all the experiments, the agents have positive, static and predefined cooperation levels. At each tick of time, the agent tries to free a constrained tile that is different from the one it is pushing into a hole, such that the sum of the costs of the paths from the agent location to the constrained tile, freeing the tile, and going back to its original position (to continue with its original task) is not larger than its cooperation level.

We first discuss the impact of the cooperative rule, presented above, for two specific scenarios (see Figure 2). These are illustrative of general ways in which the rule can generate cooperative activity. Then we present additional results gathered from using the rule in randomly generated Tileworlds.

Scenario One: The aim of each agent in this example is to fill as many holes as it can. Both agents are by design trying to get to their closest tile. In the first scenario, the tiles are at a diagonal to their final holes; since agents can't push tiles diagonally, it is more efficient for one agent to position the tile while the other pushes it without any repositioning necessary. Here, A_2 positions both tiles (getting the "assist"), while A_1 actually does the work of pushing the tiles into the holes. Total work: 11 for A_1 and 13 for A_2 (instead of 17 for each when there is no cooperative rule in force). It's important to emphasize that A_2 does not push tiles because it understands that A_1 will use them—it is simply using the cooperative rule (A_2 had a cooperation level of 2 for this scenario). The optimal solution, created perhaps by a central planner, would actually have saved the agents some work; that solution would consist of only 12 steps (5 for A_2 and 7 for A_1), but require a great deal more effort to find. In the optimal solution, A_2 does not run after tiles that A_1 eventually pushes into holes.

Scenario Two: In the second scenario, agents are blocked from their closest tiles by a barrier. As they move around the barrier, another agent prepares the target tile by pushing it to the end of the barrier. With A_1's cooperation level set at 8, A_2's level set at 4, and A_3's set at 0, the work is accomplished in 37 steps (as opposed to 48 for the non-cooperative solution). The extra work undertaken by A_1 benefits A_2; the extra work undertaken by A_2 benefits A_3. The optimal solution, more difficult to compute, takes 28 steps.

Experimental Results

We have run agent experiments on 74 randomly generated Tileworlds. All the worlds consisted of 4 agents, 6 holes, 6 tiles and 8 barriers. Each agent's primary goal is to push the closest tile into the closest hole. The worlds differed from one another by the length of the barriers (1–4) and by the locations of the agents, the holes, the tiles, and the barriers, all of which were determined randomly. We computed the number of steps that each agent carried out in pushing as many tiles as it could into the holes that were spread in an $11 * 11$ grid. The simulation stopped when there were no more tiles or holes left, or whenever the number of time ticks was 400. For each world, the agents' performance was tested with the agents being given cooperation level 0, cooperation level 1, and so on, up to and including cooperation level 8. In each of the 666 simulations (74 worlds by 9 cooperation levels), all agents were given the same cooperation level.

In 13 worlds, we found that the minimum number of steps done by the group of agents with some strictly positive cooperation level was less than the total work done by non-cooperative agents. In only 4 worlds was being cooperative actually harmful, i.e., agents cumulatively carried out more total steps to fill up the holes with any strictly positive cooperation level. In the other 57 worlds, the agents went the same number of steps when they behaved cooperatively and when they were given zero cooperation level. Therefore, in 17.56% of the worlds we tested, positive cooperation level was beneficial.

How confident can we be that this percentage reflects the real state of affairs for the overall space of worlds we were testing (i.e., 4 agents, 6 holes, etc.)? Using elementary statistical theory, we find that we can have 95% confidence that the error in probability will be less than 4.42% plus or minus for a sample size of 74 worlds (meaning we have 95% confidence that the real percentage of targeted worlds where a positive cooperation level is beneficial lies between 13.14% and 21.98%). Had we wanted to decrease the bound on the error of estimation to 2% plus or minus, we would have needed to exhaustively check 362 randomly generated worlds; to decrease the bound to 1% plus or minus, we would have needed to check 1448 worlds.[1]

The simulations were run with agents programmed to push the closest tile into the nearest hole; the chances that they would pass sufficiently close to *another* (constrained) tile to activate the cooperation rule were fairly small. Were the cooperation level sufficiently high, of course, an agent would wander far off his path to free tiles, but then it is likely that the overall performance of the group would decrease (since so much extra work is being squandered on cooperation). One can imagine other scenarios where the likelihood of finding tiles to free would be increased—for example, if the agents were sent to push arbitrary pre-assigned tiles (instead of the closest tile), and might pass other, closer, tiles on the way. In these cases, beneficial cooperation is likely to be more prevalent.

What is striking about the above, simple, experiment, is just how often a primitive, easy-to-calculate cooperative rule benefited the group as a whole. The improved performance was achieved without complicated computation or communication among the agents and the rule itself was easily identifiable for the domain. However, how would we, in general, discover suitable cooperative rules for different domains? The following section explores this question.

Cooperative Rule Taxonomy

Which kinds of rules can be designed for a given domain? Which domain characteristics are relevant for designing cooperative state-changing rules? We are interested in a general way of framing the problem of cooperative rules, that will make the analysis of a wide range of domains possible.

We define a *multi-agent deterministic finite automaton*, based on the standard definition of a deterministic finite automaton (Lewis & Papadimitriou 1981).

Definition .1 *A multi-agent deterministic finite automaton (MADFA) is a quintuple* $M = (K, \Sigma, \overline{\delta}, s, F)$*:*
- K *is a finite set of states, the set of the multi-agent world states,*
- Σ *is an alphabet. The letters are the actions that the agents can take,*
- $\overline{\delta} : K \times \overline{\Sigma} \rightarrow K$ *is the transition function.* $\overline{\Sigma}$ *denotes (multi-agent) vectors of* Σ *(i.e., multi-agent actions),*
- s *is the initial state of the multi-agent world,*
- $F \subseteq K$ *is the set of final states where all the agents' goals are achieved.*

[1] This still leaves open the question of *which* cooperation level should be used in a given world, since all we've shown is that some cooperation level is beneficial in some percentage of worlds. The optimal cooperation level may possibly be discovered through a learning algorithm, but the question remains for future research.

The language accepted by the *multi-agent deterministic finite automaton* is defined as the set of all the vector strings it accepts. For example, a word in the Tileworld domain, with three agents, could be {`{north,south,east}` `{north,west,nil}`}, `north` $\in \Sigma$, `{north, south, east}` $\in \overline{\Sigma}$. We consider two related multi-agent automata for each domain. One describes the domain in general, i.e., all the possible states and transitions that can exist in a given domain (it will be denoted by GMADFA). The second is a sub-automaton of the first, that includes only those states and transitions permitted by the agents' actual programmed behavior (that is, the sub-automaton includes only those potential transitions that might actually occur, given the way agents are programmed to act; agents may still have a choice at run-time, but the sub-automaton includes all choices they might make. It will be denoted by SMADFA.). The specific initial and final states might change for different examples, but the same architecture can be studied to find cooperative rules regardless of the details of the examples. Assuming that the rule designer has sufficient information about the domain, he can formulate these two automata that describe the domain in general and the agents' potential behavior within the domain, and use the automata to deduce appropriate cooperative rules.

The corresponding automata for two distinct domains follow:

The Tileworld SMADFA — K is the set of grid configurations of the Tileworld. $\Sigma = \{$ nil,south,east,north,west$\}$. F is the set of states in which holds ((#tiles with degree of freedom $>$ 0) $=$ 0) $\bigvee$ (#holes $=$ 0).

The FileWorld SMADFA — The FileWorld domain consists of agents whose goals are to write to and read from shared files. Whenever an agent performs an action that accesses the file, the file is locked for the other agents (i.e., they can't access it). $\Sigma = \{$*pass-lock, read, write*$\}$. One agent, having the lock, can perform the write or read action and move the world into another state in which it can continue writing or reading indefinitely. If the agent performs *pass-lock*, then the lock is passed to another agent.

The Cooperative State-Changing Rules

The purpose of a cooperative state-changing rule is to enable agents to improve the world by moving it into a state in which the agents' work will be easier. One way to make the agents' work easier is to shorten the possible paths in SMADFA leading from an initial state to a final state. A problem domain considered as an automaton can help a rule designer deduce useful cooperative rules. The rules then can be applied to dif-

ferent specific problems. For example, the cooperative rule found for the Tileworld above can be imposed on the agents in different specific scenarios. The given initial state of a particular Tileworld example doesn't matter, nor does it matter what the specific goals of each agent are; the analysis of how to improve agents' performance looks at the general actions that they can execute in the domain. We can shorten the words in three different ways (i.e., three categories of cooperative state-changing rules) by changing the SMADFA:
1. **Find a shortcut** by using the existing actions in the alphabet; i.e., look at GMADFA, at possible states and transitions, that were not included in SMADFA, and add them to it,
2. **Find a shortcut** by adding to the alphabet new actions that the agents are capable of doing,
3. **Cut loops** by minimizing the times the agents can be in a loop. We might choose to parameterize the actions; thus, cutting loops could be expressed by a change to Σ, (i.e., to the parameter that indicates the number of times the specific action can be taken).

The sociability rule presented for the Tileworld is of the first kind above—it finds a shortcut using existing actions, since the agents' original actions include the push action. Adding "extra work" to the Tileworld SMADFA means to explore other states and transitions to them such that paths from the initial state to a final state can be shorter.

Passing the lock so that other agents will also have access to a file can be a cooperative state-changing rule for the FileWorld. This rule is of the third kind; it cuts a loop created by an agent who goes on reading or writing to a file. The FileWorld SMADFA can be modified by setting a limit to the number of characters that an agent can read from or write to a file before handing over the lock.

To develop appropriate rules for different domains, and to be able to evaluate these rules, we present below some general characteristics that may prove useful in creating cooperative state-changing rules:
state dependent — a rule is *state dependent* if the extra work that needs to be done can only be accomplished in specific states. For example, in the Tileworld a tile can be freed only if there is a constrained tile and there is an agent with appropriate cooperation level that could free it. Therefore, the rule we proposed above for the Tileworld is state dependent.
guaranteed — a rule is *guaranteed* if there is certain to be no harm (no increased global work) by executing it. In the Tileworld, the rule we presented is not *guaranteed*, because the direction to which the tile is freed is heuristically computed. In the FileWorld, given that an agent has returned its lock, it is guaranteed that any other agent could use it and hence benefit from it.
reversible — a rule is *reversible* if its effects can be undone. The Tileworld rule is reversible, since any agent can push a freed tile to be next to a barrier again. In contrast, adding information to what is known by a group of agents might be irreversible.
redundant — a rule is *redundant* if performing the extra work encompassed in the rule might cause the agents to stay in the same state. Consider, for example, the StudyWorld, in which the agents are students. One of the possible actions to be performed by an agent is to borrow a book from the library. In this world, a cooperative rule might consist of a student leaving a summary of the book he has borrowed from the library. In this case, the same summary might be left again by another student, making the rule redundant.
resource dependent — a rule is *resource dependent* if following it implies the use of consumable resources (e.g., filling the printer tray with paper, although you don't have to print).

Conclusions

We have presented a "rule of cooperative behavior" that can sometimes improve overall system performance in the multi-agent Tileworld. Agents are encouraged to move tiles away from barriers (even when these tiles do not contribute to their primary goal), as long as the amount of extra work required is not too great. The addition of this implicitly cooperative behavior does not require a great deal of extra computation on the part of agents, nor any communication whatsoever. Cooperation emerges cheaply from agents acting sociably, without the overhead of negotiation or coordination. Simulations were run that illustrated the benefits of this emergent cooperation. Although unlikely to produce optimal behavior (except by chance), the cooperative rule can improve performance in a non-trivial number of instances.

We have also shown how a world can be characterized by mapping it onto an automaton. We identified three kinds of cooperative state-changing rules that can be modeled as changes to the automaton.

The principle of cooperative behavior extends to arbitrary domains, where system designers can identify aspects of global states that are generally desirable. In the Tileworld, it is generally desirable that tiles be unconstrained by barriers. In the blocks world, it is generally desirable that blocks be clear. The designers of agents can benefit by manufacturing rules of sociable behavior that encourage agents to carry out state transformations that tend to be socially desirable.

Future research will examine benefits to the system when the cooperation level of agents changes dynamically over time (for example, as a penalty mechanism aimed at uncooperative agents), how the subdivision of labor might also be affected by cooperative meta-rules, other criteria for qualifying cooperative rules, and stable sociability rules for multi-agent systems. We are also interested in looking for ananlytical ways to evaluate the cooperation level for a given domain. One way is to look at the cost of a task when the agents cooperate as a function of the cost of the original task, and to find the cooperation level that minimizes the new cost. Another way is to regard the cooperative behavior as a perturbation of the distribution of the amount of work performed by zero-cooperative agents.

References

Conry, S. E.; Meyer, R. A.; and Lesser, V. R. 1988. Multistage negotiation in distributed planning. In Bond, A. H., and Gasser, L., eds., *Readings in Distributed Artificial Intelligence.* San Mateo, California: Morgan Kaufmann Publishers, Inc. 367–384.

Davis, R. 1981. A model for planning in a multi-agent environment: steps toward principles for teamwork. Working Paper 217, Massachusetts Institute of Technology AI Laboratory.

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

Ephrati, E., and Rosenschein, J. S. 1993. Multi-agent planning as a dynamic search for social consensus. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 423–429.

Ferber, J., and Drogoul, A. 1992. Using reactive multi-agent systems in simulation and problem solving. In Avouris, N. M., and Gasser, L., eds., *Distributed Artificial Intelligence: Theory and Praxis.* Kluwer Academic Press. 53–80.

Genesereth, M. R.; Ginsberg, M. L.; and Rosenschein, J. S. 1986. Cooperation without communication. In *Proceedings of the National Conference on Artificial Intelligence*, 51–57.

Gmytrasiewicz, P., and Durfee, E. H. 1992. A logic of knowledge and belief for recursive modeling: Preliminary report. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 628–634.

Grosz, B., and Kraus, S. 1993. Collaborative plans for group activities. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 367–373.

Hanks, S.; Pollack, M. E.; and Cohen, P. R. 1993. Benchmarks, test beds, controlled experimentation, and the design of agent architectures. *AI Magazine* 17–42.

Kraus, S., and Wilkenfeld, J. 1991. Negotiations over time in a multi agent environment: Preliminary report. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, 56–61.

Kraus, S. 1993. Agents contracting tasks in non-collaborative environments. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 243–248.

Kreifelts, T., and Martial, F. 1990. A negotiation framework for autonomous agents. In *Proceedings of the Second European Workshop on Modeling Autonomous Agents and Multi-Agent Worlds*, 169–182.

Kuwabara, K., and Lesser, V. R. 1989. Extended protocol for multistage negotiation. In *Proceedings of the Ninth Workshop on Distributed Artificial Intelligence*, 129–161.

Lewis, H. R., and Papadimitriou, C. H. 1981. *Elements of the theory of computation.* Prentice-Hall, Inc.

Malone, T.; Fikes, R.; and Howard, M. 1988. Enterprise: A market-like task scheduler for distributed computing environments. In Huberman, B. A., ed., *The Ecology of Computation.* Amsterdam: North-Holland Publishing Company. 177–205.

McArthur, D.; Steeb, R.; and Cammarata, S. 1982. A framework for distributed problem solving. In *Proceedings of the National Conference on Artificial Intelligence*, 181–184.

Montgomery, T. A.; Lee, J.; Musliner, D. J.; Durfee, E. H.; Damouth, D.; and So, Y. 1992. *MICE Users Guide.* Artificial Intelligence Laboratory, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, Michigan.

Pollack, M. E., and Ringuette, M. 1990. Introducing the Tileworld: Experimentally evaluating agent architectures. In *Proceedings of The National Conference on Artificial Intelligence*, 183–189.

Shoham, Y., and Tennenholtz, M. 1992a. Emergent conventions in multi-agent systems: initial experimental results and observations (preliminary report). In *Principles of knowledge representation and reasoning: Proceedings of the Third International Conference(KR92).*

Shoham, Y., and Tennenholtz, M. 1992b. On the synthesis of useful social laws for artificial agent societies (preliminary report). In *Proceedings of the Tenth National Conference on Artificial Intelligence.*

Smith, R. G. 1978. *A Framework for Problem Solving in a Distributed Processing Environment.* Ph.D. Dissertation, Stanford University.

Sycara, K. 1988. Resolving goal conflicts via negotiation. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 245–250.

Sycara, K. 1989. Argumentation: Planning other agents' plans. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 517–523.

Tennenholtz, M., and Moses, Y. 1989. On cooperation in a multi-entity model. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 918–923.

Zlotkin, G., and Rosenschein, J. S. 1993a. Compromise in negotiation: Exploiting worth functions over states. Technical Report 93–3, Leibniz Center for Computer Science, Hebrew University.

Zlotkin, G., and Rosenschein, J. S. 1993b. A domain theory for task oriented negotiation. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 416–422.