A Fast First-Cut Protocol for Agent Coordination

Andrew P. Kosoresow*
Department of Computer Science
Stanford University
Stanford, CA 94305, U.S.A.
kos@theory.stanford.edu

Abstract

This paper presents a fast probabilistic method for coordination based on Markov processes, provided the agents' goals and preferences are sufficiently compatible. By using Markov chains as the agents' inference mechanism, we are able to analyze convergence properties of agent interactions and to determine bounds on the expected times of convergence. Should the agents' goals or preferences not be compatible, they can detect this situation since coordination has not been achieved within a probabilistic time bound and the agents can then resort to a higher-level protocol. The application, used for motivating the discussion, is the scheduling of tasks, though the methodology may be applied to other domains. Using this domain, we develop a model for coordinating the agents and demonstrate its use in two examples.

Introduction

In distributed artificial intelligence (DAI), coordination, cooperation, and negotiation are important in many domains. Agents need to form plans, allocate resources, and schedule actions, considering not only their own preferences, but also those of other agents with whom they have to interact. Making central decisions or deferring to another agent may not be possible or practical, because of design constraints or political considerations. In these situations, agents will need some mechanism for coming to an agreement without reference to an outside authority. There are other advantages to having a distributed negotiator. Communication patterns may become more balanced when a central node or set of nodes do not have to participate in every interaction. Information is localized. Each person's information is only contained by the local agent and can be more closely controlled. Further, as each person's (or set of persons') schedule is maintained by a separate agent, the system would degrade gracefully if some of the agents were to go off line. Given that there exists at least one satisfactory agreement that satisfies all the agents' constraints, we want to find such an agreement within a reasonable amount of time. If such an agreement does not exist, we would like to find an approximate agreement satisfying as many constraints as possible.

In this paper, we propose a probabilistic method using Markov processes for the coordination of agents, using the domain of scheduling tasks. We make two assumptions about the capabilities of the agents: Agents have a planning system capable of generating sets of possible plans and they have a high-level negotiation system capable of exchanging messages about goals, tasks, and preferences. Each agent has a set of tasks that it has to accomplish. Some of these tasks require the participation of other agents. The agents may have some preference for who does which tasks. While it is possible for the agents to enter into full-scale negotiations immediately, we propose to have the agents first go through a brief phase of trading offers and counteroffers. Should the agents' goals and preferences be sufficiently compatible, the agents will come to an agreement with high probability without the need for full-scale negotiation. Otherwise, the agents would realize this and resort to the higher-level negotiation protocol.[1]

We propose that the agents would simultaneously post their proposed schedules. Based on these postings, the agents would compute their next proposal and repost until they had synchronized on a schedule. In order to calculate each posting, each agent has a Markov process generating its next schedule based

*This research was supported jointly by the National Aeronautics and Space Administration under grant NCC2-494-S11, the National Science Foundation under grant IRI-9116399, and the Rockwell International Science Center–Palo Alto Facility.

[1] If the high-level protocol takes time T, the low-level protocol takes time t, and the low-level protocol succeeds some fraction p of the time for a set of k tasks, then preprocessing is worthwhile if

$$kT \leq p(kt) + (1-p)(kT)$$

on the current postings of all the agents; this process would involve a lookup operation and possibly a random coin flip. By combining the Markov processes of the individual agents, we obtain a Markov chain for the entire system of agents. We can then analyze the properties of the system by analyzing the Markov chain and determine if the agents will converge to an agreeable schedule and the expected time of convergence.

If the agents have a model of the other agents' tasks and preferences, they can then conjecture a Markov chain corresponding to the entire system of agents. Given this information, the agent can estimate the number of iterations needed to achieve coordination for a given probability. If coordination has not been achieved by this time, the agent then knows that its model of the other agents was incorrect (or that it has been very unlucky) and can then use the higher-level protocol.

In the rest of the paper, we will develop a methodology for using Markov processes and give several illustrative examples of how task scheduling would work using Markov processes. In the next section, we give some basic definitions and lay the groundwork for the use of Markov processes as a method of negotiation. Next, we define convergence properties and expected times. In the following section, we give two examples in a task scheduling domain demonstrating the use of Markov processes. Finally, we discuss related work and summarize our current results.

Basic Architecture

In this paper, we consider a scenario where a group of agents needs to come to an agreement. We assume that the agents communicate by posting messages simultaneously at discrete times called *clock ticks* and that the posted messages are visible to all the agents concerned.[2] At a clock tick, each agent is required to post some message. If it fails to do so for some reason, a default will be specified whose definition depends on the application. First let us consider the process of coming to an agreement from the perspective of an individual agent and then examine the process from the perspective of the entire system of agents.

After each clock tick, an individual agent in the system can examine the current postings of all the agents for that clock tick. Based on this information, the agent needs to generate its proposed schedule by the next clock tick. One method for generating a schedule under this constraint is to use Markov procedures. Informally, a *Markov procedure* is a function that generates an action taking only the agent's current world state as the input.[3] Since many rounds of negotiation may be necessary to come to a satisfactory agreement, Markov procedures are promising candidates for use in such a system; for each iteration, all that is potentially needed is a single lookup or a simple computation.

Each agent is controlled by a *Markov process*, a variant of a Markov procedure. A Markov process is a function that takes the system state and a randomly-generated number and yields an action. In our model, the *system state* is the set of offers made by the agents during the last clock tick. The action is the agent's offer for the next clock tick. Thus, given the set of all offers, $\mathcal{O}$, and a system state, $\mathcal{S}$, we have an evaluation function, $\mathcal{E}_{\mathcal{S}}$, such that given an $o \in \mathcal{O}$, $\mathcal{E}_{\mathcal{S}}(o)$ is the probability that the agent should offer o in system state $\mathcal{S}$ and $\sum_{o \in \mathcal{O}} \mathcal{E}_{\mathcal{S}}(o) = 1$. Let $\mathcal{A}_i^t$ be the state of the i^{th} agent at time t. So, given a random number generator, we can define a Markov process, $\mathcal{M}$ to generate each agent's next offer based on $\mathcal{E}_{\mathcal{S}}$. Given that there are n agents and ρ is a random number such that $0 \leq \rho \leq 1$, $\mathcal{A}_i^{t+1} = \mathcal{M}_i(\mathcal{A}_1^t, \mathcal{A}_2^t, \ldots, \mathcal{A}_n^t, \rho)$ for the i^{th} agent. A starting state is specified by each of the agents either by choosing an action randomly or by choosing a preferred action based on some criteria. Practically speaking, the entire Markov process for an agent will probably never be explicitly specified as it could contain an exponential number of states. For example, the number of possible ways to allocate a set of tasks among a group of agents is exponential in the number of tasks. More likely, the system states will be divided into equivalence classes, for which only a single set of actions will be considered. Furthermore, the agents' constraints and preferences may preclude them from even considering many possible ways of allocating the tasks. For an effective system, the number of these classes should be sub-exponential. We shall see an example of how system states collapse into a fewer number of equivalent classes.

In order to learn something of the properties of this system, we can consider all the agents and their Markov processes as a single Markov chain.[4] We call this the *System Markov Chain* or simply the *system chain*. Using the above notation, the system process, $\mathcal{SM}$, can be represented as $\mathcal{S}^{t+1} = \mathcal{SM}(\mathcal{S}^t, P)$ where $\mathcal{S}^t = (\mathcal{A}_1^t, \mathcal{A}_2^t, \ldots, \mathcal{A}_n^t)$ and P is the n-tuple of ρs, as defined above for Markov processes. As it is a Markov chain, certain properties including convergence and the expected times of convergence may be computed in many cases.

[2] These assumptions provide a model that is simpler to analyze and may be later extended to cover agent-to-agent messages. They also allow us to postpone considering agents that are Byzantine and otherwise malicious. For example, consider playing a game of Stone, Scissors, Paper. If one player delays slightly, it can win with knowledge of the the opponent's move. Similarly, if the players are communicating through an intermediary, the intermediary can distort the game by passing false information or delaying it.

[3] While ignoring most of the history of an agent may seem to be restrictive, it has been shown to be effective, for example, in cases such as the Tit-for-Tat Algorithm for the Prisoners' Dilemma [Axelrod, 1984].

[4] See [Freedman, 1983] for a more complete discussion of Markov chains and related topics.

Convergence and Expected Times

We now define whether the system chain converges or, in other words, whether the agents come to an agreement. First we need to define absorbing states for a system; a state $\mathcal{S}$ is defined to be an *absorbing state* if and only if given the system is in state $\mathcal{S}$ at time t, it will be in state $\mathcal{S}$ at time $t+1$ with probability 1. If a Markov chain has at least one absorbing state, it is called an *absorbing Markov chain*. If an absorbing Markov chain has a non-zero probability of reaching some absorbing state from every state, let us call it a *converging Markov chain*, since eventually the Markov chain will end up in one of these states. If the system chain is a converging Markov chain and all the absorbing states are possible valid agreements, the agents' Markov processes will lead to one of these agreements. Let us call this chain a *converging system chain*. In this case, a *valid agreement* is one where either all the agents' constraints are satisfied or, if that is not possible, a satisfactory approximate solution is reached. Thus, if we can have the agents generate Markov processes which lead to a converging system chain, we know that they will eventually come to an agreement.

In our case, the convergent states consist of those where all of the agents issue the same offer during one time-step and the subsequent states for all the agents is that same state. We can take advantage of partially convergent states when agents agree on the assignment of subsets of their tasks. If agreeing on these tasks does not preclude reaching an absorbing state, then the agents can reduce their search-spaces, by fixing the agreed-upon task assignments.

We can also try to figure out the expected convergence time. Given a Markov chain $\mathcal{M}$ and a distribution $\mathcal{D}$ of initial system states, the *expected time of convergence*, $\mathcal{T}(\mathcal{M},\mathcal{D})$ is defined to be:

$$\mathcal{T}(\mathcal{M},\mathcal{D}) = \sum_{i=0}^{\infty} i P_c(\mathcal{M},\mathcal{D},i)$$

where $P_c(\mathcal{M},\mathcal{D},t)$ is the probability that the Markov chain $\mathcal{M}$ will reach an absorbing state at time t. This latter quantity can be computed by either solving the recurrence relations derived from the specification of the Markov chain or by framing the Markov chain as a network-flow problem and simulating it. Thus, proving convergence and calculating the time of convergence for the system chain will let us know whether an agreement is guaranteed and, if so, approximately when.

The Scheduling of Multiple Tasks

In this section, we show how to apply Markov processes to scheduling tasks. We will give two examples and an analysis of their expected times of convergence.

Each of the agents in the system has some set of tasks that it has to schedule, possibly the empty set.

1. Exchange task lists with other agents.
2. Using task lists, form a Markov process to generate schedule offers either randomly or weighted according to the agent's preferences or constraints.
3. Form Markov chain using the agent's Markov process and the agent's estimates of the others' Markov processes. Determine estimated time of convergence using the Markov chain.
4. Run Markov process until the time bound, established in Step 3, is reached or a suitable agreement is found.
5. If an agreement is found, return the agreement. Otherwise, resort to higher-level protocol.

Figure 1: Outline of the first-cut coordination protocol.

Assume that there are n agents and they are concerned with a set of m available time slots. For the rest of the paper, we assume that the tasks take unit time and are independent. Each of the tasks requires some number of agents to complete it.

Initially, the agents trade lists of tasks that need to be assigned and the available times for the tasks. By not communicating constraints and preferences in this protocol, agents may avoid having to calculate, communicate, and reason with these types of facts. Each of the agents then generates a Markov process $\mathcal{M}$, as defined above, based on the set of task schedules that it considers valid. Each of the schedules consists of sets of the agents' tasks that should be done during a particular time slot. Thus, the tuple $(\{Aa, Bb\}, \{Bc\}, \{Aa, Bd\})$ indicates that the agent suggests that the agents A and B do tasks a and b in time 1 respectively, B does task c in time 2, and A and B do tasks a and d in time 3 respectively. Since the agent does not know the other agents' constraints, some or all of the schedules generated by one agent may not be valid for other agents. If the agent has some information about the other agents' constraints (or it is willing to make some assumptions in the absence of this information), it can now form a system chain. By calculating the expected time of convergence for the system chain, it now has an estimate on how long the protocol will take and it can use that information to decide when to go to the higher-level protocol. The agents will then trade offers until an agreement is reached or a deadline is passed. This process is outlined in Figure 1.

For this system, there is an exponential number of possible schedules for tasks, and thus both the agents' Markov processes and the system chain could have exponential size if they were represented explicitly. However, often it is possible to store the agents' Markov processes implicitly. Further, we can reduce the number of system chain states by defining equivalence classes over the system states. For example, the set of states where everyone has agreed on a schedule can be dealt with as a single class. In the following example, we show how this fact can be used and give an

System state names:		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
Agent A's action equivalence classes:		e	b	d	e	d	e	a	c	c	a	e	d	e	d	b	e
System state equivalence class:		1	2	2	1	2	1	3	2	2	3	1	2	1	2	2	1
Agent states at time t:	Agent A:	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
	Agent B:	1	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2
	Agent C:	1	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2
	Agent D:	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
Corresponding agent actions at time $t+1$:	Agent A	R	2	1	R	1	R	1	1	2	2	R	2	R	2	1	R
	Agent B:	R	1	2	R	2	R	2	2	1	1	R	1	R	1	2	R
	Agent C:	R	1	2	R	2	R	2	2	1	1	R	1	R	1	2	R
	Agent D:	R	2	1	R	1	R	1	1	2	2	R	2	R	2	1	R

Read system states down vertically.

Figure 2: Markov processes for two teams of two agents coordinating usage of a resource. ('R' in the table above corresponds to a equiprobable random choice between '1' and '2'.)

overview of the procedure. In the second example, we sketch a case where the agents' constraints are partially incompatible.

Example 1: A Constrained Resource

Suppose there are 4 agents: A, B, C, and D. There are two tasks that need to be done: t_1 and t_2. Let A and D be the team that has to do t_1 and let B and C be the team that has to do t_2. Finally, assume that both tasks utilize some resource that can only be used by one set of agents at a time and there are two time slots available: 1 and 2.[5] This example could represent two teams of agents having to use a particular tool in a job-shop or two sets of siblings having to share a television set. Assuming that the agents find the resource equally desirable during both time slots, and that all the agents have equal input into the decision, we need to decide an order in which the agent teams get to use the resource. While workers in a job-shop or family might have other means for coming to a decision, we can abstract the situation and use Markov processes to come to a decision. The agents can specify their choice by a '1' or a '2'. '1' will indicate that the agent offers to do its task first, while a '2' indicates that it offer to go second. Thus, we design the Markov process shown in Figure 2 for each of the agents.

In this Markov process, each of the agents has to react to one of sixteen possible system states. As shown in Figure 2 for Agent A, each of the states falls into one of five equivalence classes for the agents, each of which corresponds to an action: (a) the agents are coordinated; (b) the agents are almost coordinated and to achieve coordination, I need to flip state; (c) the agents are almost coordinated and to achieve coordination, the other person in my pair needs to flip state; (d) the agents are almost coordinated and to achieve coordination, one of the people in the other pair needs

[5] A simpler example consists of two agents trying to coordinate on a coin, where both of them decide on heads or tails. This is an example of a consensus problem as in [Aspnes and Waarts, 1992]. The solution for the given example can be easily modified to give a solution to multi-agent consensus.

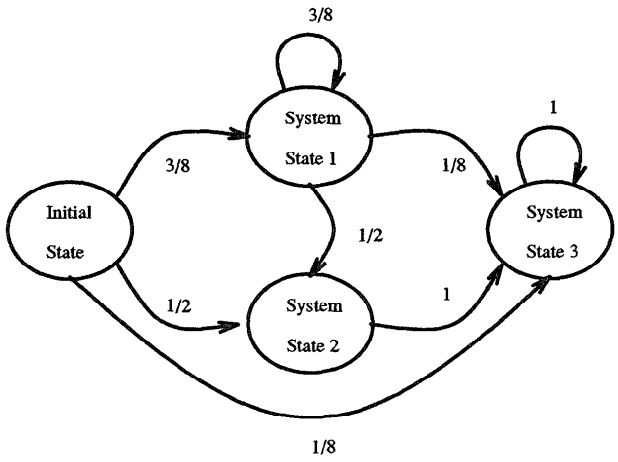

Figure 3: The resulting system chain for Example 1. (State labels correspond to system state equivalence classes from Figure 2 and arc labels are probabilities of transitioning between the two given states.)

to flip state; and (e) the agents are uncoordinated and need to flip randomly. Similarly, the system states fall into three equivalence classes: (1) the agents are uncoordinated, (2) the agents are almost coordinated (and will get coordinated in the next time step), and (3) the agents are coordinated and need to flip randomly. These three equivalence classes are used to form the system chain shown in Figure 3. This Markov chain converges and the expected time to reach a convergent state is $\frac{8}{5}$.[6]

In this example, using Markov processes leads to a satisfactory solution in a relatively short expected time. The solution is fair since both outcomes are equally probable and no agent had more influence than any other agent. While there is no deterministic time bound for the agents coming to an agreement, there is no chance of deadlock which may result from a deterministic protocol. Further, we do not depend on ordering the agents or on an external device for insuring honesty. Of course, there are situations where the agents can sabotage the efficiency or the convergence of such a method. Suppose that Agent D has decided that it prefers the first time slot. It can raise the probability of choosing that slot to greater than $\frac{1}{2}$ or even go to the extreme of choosing it no matter what, thus increasing the expected time of convergence. Further, if Agent A decides that it must have the second time slot, we are left in a situation that has no satisfactory schedule. While such situations appear to lead to inefficiency or inability to come to a satisfactory solution, they may be used to take preferences into account and discover situations where there are irreconcilable conflicts among the agents. An agent can be probabilisti-

[6] The details of the expected time calculations is left to the full-length version of this paper.

cally certain that something unexpected is happening when the system has not converged to an agreement within a time bound.

If the agents have a model of each other, then they can construct the system chain for their particular case. Given the chain, the agents can either analytically determine the time of convergence for simple chains or derive it empirically for more complex ones. Similarly, they can use the cutoff for negotiation as an indication that they are not making progress. For example, if they assume the model described above, they would converge to a solution over 90% of the time within seven iterations. Thus, they can use seven as a cutoff to indicate that there might be incompatible preferences such as the ones described above for Agents A and D.

Example 2: Non-compatible Goals

In the second example, we have two agents whose goals are not sufficiently compatible. Even though coordination failures may occur in this system, there is still a significant probability that the agents will coordinate on a plan and thus will not have to resort to a higher level protocol. It further demonstrates how convergence on a partial solution can lead to fast convergence during the remainder of the protocol.

Suppose there are two agents, A and B. Their goal is to move a set of objects to one of three locations: 1, 2, or 3. Agent A would like to have objects a, b, and c at the same location, and Agent B would like to have objects b, c, and d at the same location. There are twelve possible tasks consisting of moving the four objects to the three locations, such as task $a1$ which consists of moving object a to location 1. The agents each have two time steps available during which they can schedule their actions. The initial communications would contain the tasks each agent needs done and the time frame. For example, Agent A would send $((a1, b1, c1, a2, b2, c2, a3, b3, c3), (T1, T2))$ indicating the nine tasks that it is interested in and the two time steps. Note that only the tasks were communicated without any reference to constraints or preferences. The computation and communication of this information is left to the higher-level protocol, should it be necessary.

Since we assume that tasks are of unit time and do not depend on each other, each agent can just enter the new tasks into its list of things to do and expand its Markov process to include them as a possibility. If this operation is not straightforward, as it is in this case, the agents might pop out of this low-level protocol and proceed with a higher-level one at this point.

The agents' offers would consist of the possible ways to do their tasks combined with the possible ways they believe that the other agent's task could be incorporated. For example, Agent A's possible offers, at this point, consists of 216 possibilities: Each of the tasks needs to be assigned an agent, a time, and a location with the constraint that a, b, and c are assigned the same location. These can be represented by each of the agents implicitly. At each time step, the agents would suggest one of these offers and then compare to see they have made any progress.

At this point, the agents can take advantage of partial coordination. If an assignment of tasks to agents, tasks to time slots, or tasks to locations has been agreed on, the agents then fix those parameters and accordingly reduce the number of offers that they would consider for the next round. Let us first consider the assignment of the objects to locations. There are 81 possible outcomes in this case. With probability $\frac{1}{27}$ the agents will coordinate on a possible assignment of locations; with probability $\frac{2}{27}$, the agents will result in a state from which an acceptable plan cannot be reached; with probability $\frac{8}{27}$, the agents will still be in the initial state; and with $\frac{16}{27}$, the agents will be partially coordinated where the probability is better than $\frac{1}{2}$ that they will eventually agree on an acceptable plan. Thus, the termination of this part of the schedule has an expected time of at most nine steps.

In the other portion of the schedule, where we assign tasks to agents, there is no chance of a partial solution leading to a deadlock state. The agents simply agree on how to put four tasks in four slots. Any partial solutions in this part of the schedule will constrain subsequent actions leading to an agreement.

Thus we see several important features of this methodology. The protocol is able to restrict the space of possible schedules by utilizing partial agreements. Even in an unfavorable situation, it will lead to an acceptable schedule in more than half the cases. Further, we were able to use this coordination protocol even though the two agents were unaware of the constraints of the other agents. By using a quick offer-counteroffer protocol, there would be no need to trade this information and to do the computations associated with incorporating the information into the planner and/or the higher-level coordination protocol's data structures. Even if our protocol was unsuccessful, it may provide us with some useful information for the higher-level protocol.

Summary of Current and Related Work

In this paper, we draw on a variety of sources. Previous work in coordination and cooperation includes the game-theoretic approach taken by Genesereth, Rosenschein, and Ginsberg as in [Ginsberg, 1985], for example. One advantage of our method is that knowledge of explicit payoff matrices is not necessary. Our method is also applicable to the more current work on the Postman Problem[Zlotkin and Rosenschein, 1989]. Our method is more similar contract nets[Davis and Smith, 1983], though our approach is more like haggling than bidding for a contract.

In [Ephrati and Rosenschein, 1991], [Ephrati and Rosenschein, 1992], [Jennings and Mamdani, 1992],

and [Wellman, 1992], higher-level protocols are proposed for agent coordination and negotiation. These may be suitable for use with our protocol in part or in their entirety. Other work is related to providing components for a system for coordination and negotiation. For example, [Gmytrasiewicz *et al.*, 1991] provides a framework whereby agents can build up a data structure describing other agents with whom they interact, and [Kraus and Wilkenfeld, 1991] provides a framework for incorporating time as part of a negotiation. Our protocol might also be adapted for use in multi-agent systems such as those described in [Shoham and Tennenholtz, 1992].

Taking these considerations into account, we have decided to use a probabilistic method with Markov processes and chains to try to guarantee that the agents will come to some agreement; the randomness provides us a tool for avoiding potential deadlock among the processes. Further, using Markov processes provides us with methods for determining whether a system of agents will converge to an agreement and, if so, in what expected time. We have demonstrated how Markov processes and Markov chains can be used for coordinating tasks. These techniques may also be useful for other domains involving coordination or negotiation.

We are currently looking at applying Markov processes to the scheduling of elevators, job shops, and other resource allocation problems. We are also working on implementing these procedures to test them empirically. There are also several general problems that need to be examined. In order to make this technique more convenient, we need to look at how to generate Markov chains from a formal specification of a problem and how to recognize equivalence classes. To make the method less restrictive, it would be useful to try to relax the requirement that messages be posted simultaneously. Further, it might be useful to employ direct messages between agents instead of having them post their messages. Finally, we would like to incorporate profit/cost metrics and more complex interdependent tasks with temporal or ordering constraints. As it stands, we see Markov processes as a useful technique for exploring agent coordination and, subsequently, negotiation.

Acknowledgements

I would like to thank Nils Nilsson and Narinder Singh, for their many valuable comments and discussions regarding this paper. In addition, I would like to thank Anil Gangolli, Matt Ginsberg, Michael Genesereth, Andrew Goldberg, Andrew Golding, Ramsey Haddad, Jane Hsu, Joseph Jacobs, George John, Illah Nourbakhch, Oren Patashnik, Greg Plaxton, Devika Subramanian, Vishal Sikka, Eric Torng, Rich Washington, James Wilson, and the members of the Bots, MUGS, and Principia Research Groups at Stanford for many productive conversations.

References

Aspnes, James and Waarts, Orli 1992. Randomized consensus in expected $O(n \log^2 n)$ operations per processor. In *33rd Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press. 137–146.

Axelrod, Robert 1984. *The Evolution of Cooperation*. Basic Books, Inc., New York.

Davis, Randall and Smith, Reid G. 1983. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence* 20:63–109.

Ephrati, Eithan and Rosenschein, Jeffrey S. 1991. The clarke tax as a consensus mechanism among automated agents. In *Proceedings of the Ninth National Conference on Artificial Intelligence*. Morgan Kaufmann. 173–178.

Ephrati, Eithan and Rosenschein, Jeffrey S. 1992. Constrained intelligent action: Planning under the influence of a master agent. In *Proceedings of the Tenth National Conference on Artificial Intelligence*. Morgan Kaufmann. 263–268.

Freedman, David 1983. *Markov Chains*. Springer-Verlag.

Ginsberg, Matthew L. 1985. Decision procedures. In Huhns, Michael N., editor 1985, *Distributed Artificial Intelligence*. Morgan Kaufman. chapter 1, 3–28.

Gmytrasiewicz, Piotr J.; Durfee, Edmund H.; and Wehe, David K. 1991. The utility of communication in coordinating intelligent agents. In *Proceedings of the Ninth National Conference on Artificial Intelligence*. Morgan Kaufmann. 166–172.

Jennings, N. R. and Mamdani, E. H. 1992. Using joint responsibility to coordinate collaborative problem solving in dynamic environments. In *Proceedings of the Tenth National Conference on Artificial Intelligence*. Morgan Kaufmann. 269–275.

Kraus, Sarit and Wilkenfeld, Jonathan 1991. The function of time in cooperative negotiations. In *Proceedings of the Ninth National Conference on Artificial Intelligence*. Morgan Kaufmann. 179–184.

Shoham, Yoav and Tennenholtz, Moshe 1992. Emergent conventions in multi-agent systems: initial experimental results and observations. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*. Morgan Kaufmann. 225–231.

Wellman, Michael P. 1992. A general-equilibrium approach to distributed transportation planning. In *Proceedings of the Tenth National Conference on Artificial Intelligence*. Morgan Kaufmann. 282–289.

Zlotkin, Gilad and Rosenschein, Jeffrey S. 1989. Negotiation and task sharing among autonomous agents in cooperative domains. In *Eleventh International Joint Conference on Artificial Intelligence*. Morgan Kaufmann. 912–917.