Learning to coordinate without sharing information

Sandip Sen, Mahendra Sekaran, and John Hale
Department of Mathematical & Computer Sciences
University of Tulsa
600 South College Avenue
Tulsa, OK 74104-3189
sandip@kolkata.mcs.utulsa.edu

Abstract

Researchers in the field of Distributed Artificial Intelligence (DAI) have been developing efficient mechanisms to coordinate the activities of multiple autonomous agents. The need for coordination arises because agents have to share resources and expertise required to achieve their goals. Previous work in the area includes using sophisticated information exchange protocols, investigating heuristics for negotiation, and developing formal models of possibilities of conflict and cooperation among agent interests. In order to handle the changing requirements of continuous and dynamic environments, we propose learning as a means to provide additional possibilities for effective coordination. We use reinforcement learning techniques on a block pushing problem to show that agents can learn complimentary policies to follow a desired path without any knowledge about each other. We theoretically analyze and experimentally verify the effects of learning rate on system convergence, and demonstrate benefits of using learned coordination knowledge on similar problems. Reinforcement learning based coordination can be achieved in both cooperative and non-cooperative domains, and in domains with noisy communication channels and other stochastic characteristics that present a formidable challenge to using other coordination schemes.

Introduction

In this paper, we will be applying recent research developments from the reinforcement learning literature to the coordination problem in multiagent systems. In a reinforcement learning scenario, an agent chooses actions based on its perceptions, receives scalar feedbacks based on past actions, and is expected to develop a mapping from perceptions to actions that will maximize feedbacks. Multiagent systems are a particular type of distributed AI system (Bond & Gasser 1988), in which autonomous intelligent agents inhabit a world with no global control or globally consistent knowledge. These agents may still need to coordinate their activities with others to achieve their own local goals. They could benefit from receiving information about what others are doing or plan to do, and from sending them information to influence what they do.

Coordination of problem solvers, both selfish and cooperative, is a key issue to the design of an effective distributed AI system. The search for domain-independent coordination mechanisms has yielded some very different, yet effective, classes of coordination schemes. Almost all of the coordination schemes developed to date assume explicit or implicit sharing of information. In the explicit form of information sharing, agents communicate partial results (Durfee & Lesser 1991), speech acts (Cohen & Perrault 1979), resource availabilities (Smith 1980), etc. to other agents to facilitate the process of coordination. In the implicit form of information sharing, agents use knowledge about the capabilities of other agents (Fox 1981; Genesereth, Ginsberg, & Rosenschein 1986) to aid local decision-making. Though each of these approaches has its own benefits and weaknesses, we believe that the less an agent depends on shared information, and the more flexible it is to the on-line arrival of problem-solving and coordination knowledge, the better it can adapt to changing environments.

In this paper, we discuss how reinforcement learning techniques of developing policies to optimize environmental feedback, through a mapping between perceptions and actions, can be used by multiple agents to learn coordination strategies without having to rely on shared information. These agents, though working in a common environment, are unaware of the capabilities of other agents and may or may not be cognizant of goals to achieve. We show that through repeated problem-solving experience, these agents can develop policies to maximize environmental feedback that can be interpreted as goal achievement from the viewpoint of an external observer. This research opens up a new dimension of coordination strategies for multiagent systems.

Acquiring coordination knowledge

Researchers in the field of machine learning have investigated a number of schemes for using past experience to improve problem solving behavior (Shavlik & Dietterich 1990). A number of these schemes can be effectively used to aid the problem of coordinating multiple agents inhabiting a common environment. In cooperative domains, where agents have approximate models of the behavior of other agents and are willing to reveal information to enable the group perform better as a whole, pre-existing domain knowledge can be used inductively to improve performance over time. On the other hand, learning techniques that can be used incrementally to develop problem-solving skills relying on little or no pre-existing domain knowledge can be used by both cooperative and non-cooperative agents. Though the latter form of learning may be more time-consuming, it is generally more robust in the presence of noisy, uncertain, and incomplete information.

Previous proposals for using learning techniques to coordinate multiple agents have mostly relied on using prior knowledge (Brazdil et al. 1991), or on cooperative domains with unrestricted information sharing (Sian 1991). Even previous work on using reinforcement learning for coordinating multiple agents (Tan 1993; Weiß1993) have relied on explicit information sharing. We, however, concentrate on systems where agents share no problem-solving knowledge. We show that although each agent is independently optimizing its own environmental reward, global coordination between multiple agents can emerge without explicit or implicit information sharing. These agents can therefore act independently and autonomously, without being affected by communication delays (due to other agents being busy) or failure of a key agent (who controls information exchange or who has more information), and do not have to be worry about the reliability of the information received (Do I believe the information received? Is the communicating agent an accomplice or an adversary?). The resultant systems are, therefore, robust and general-purpose.

Reinforcement learning

In reinforcement learning problems (Barto, Sutton, & Watkins 1989; Holland 1986; Sutton 1990), reactive and adaptive agents are given a description of the current state and have to choose the next action from a set of possible actions so as to maximize a scalar *reinforcement* or *feedback* received after each action. The learner's environment can be modeled by a discrete time, finite state, Markov decision process that can be represented by a 4-tuple $\langle S, A, P, r\rangle$ where $P : S \times S \times A \mapsto [0, 1]$ gives the probability of moving from state s_1 to s_2 on performing action a, and $r : S \times A \mapsto \Re$ is a scalar reward function. Each agent maintains a policy, π, that maps the current state into the desirable action(s) to be performed in that state. The expected value of a discounted sum of future rewards of a policy π at a state x is given by $V_\gamma^\pi \stackrel{\text{def}}{=} E\{\sum_{t=0}^{\infty} \gamma^t r_{s,t}^\pi\}$, where $r_{s,t}^\pi$ is the random variable corresponding to the reward received by the learning agent t time steps after if starts using the policy π in state s, and γ is a discount rate ($0 \leq \gamma < 1$).

Various reinforcement learning strategies have been proposed using which agents can can develop a policy to maximize rewards accumulated over time. For our experiments, we use the Q-learning (Watkins 1989) algorithm, which is designed to find a policy π^* that maximizes $V_\gamma^\pi(s)$ for all states $s \in S$. The decision policy is represented by a function, $Q : S \times A \mapsto \Re$, which estimates long-term discounted rewards for each state–action pair. The Q values are defined as $Q_\gamma^\pi(s, a) = V_\gamma^{a;\pi}(s)$, where $a;\pi$ denotes the event sequence of choosing action a at the current state, followed by choosing actions based on policy π. The action, a, to perform in a state s is chosen such that it is expected to maximize the reward,

$$V_\gamma^{\pi^*}(s) = \max_{a \in A} Q_\gamma^{\pi^*}(s, a) \text{ for all } s \in S.$$

If an action a in state s produces a *reinforcement* of R and a transition to state s', then the corresponding Q value is modified as follows:

$$Q(s, a) \leftarrow (1-\beta)\, Q(s, a) + \beta\, (R + \gamma \max_{a' \in A} Q(s', a')). \quad (1)$$

Block pushing problem

To explore the application of reinforcement learning in multi-agent environments, we designed a problem in which two agents, a_1 and a_2, are independently assigned to move a block, b, from a starting position, S, to some goal position, G, following a path, P, in Euclidean space. The agents are not aware of the capabilities of each other and yet must choose their actions individually such that the joint task is completed. The agents have no knowledge of the system physics, but can perceive their current distance from the desired path to take to the goal state. Their actions are restricted as follows; agent i exerts a force $\vec{F}_i$, where $0 \leq |\vec{F}_i| \leq F_{max}$, on the object at an angle θ_i, where $0 \leq \theta \leq \pi$. An agent pushing with force $\vec{F}$ at angle θ will offset the block in the x direction by $|\vec{F}|\cos(\theta)$ units and in the y direction by $|\vec{F}|\sin(\theta)$ units. The net resultant force on the block is found by vector addition of individual forces: $\vec{F} = \vec{F}_1 + \vec{F}_2$. We calculate the new position of the block by assuming unit displacement per unit force along the direction of the resultant force. The new block location is used to provide *feedback* to the agent. If (x, y) is the new block location, $P_x(y)$ is the x-coordinate of the path P for the same y coordinate, $\Delta x = |x - P_x(y)|$ is the distance along the x dimension between the block and the desired path, then $K * a^{-\Delta x}$ is the feedback given to each agent for their last action (we have used $K = 50$ and $a = 1.15$).

The field of play is restricted to a rectangle with endpoints $[0, 0]$ and $[100, 100]$. A trial consists of the agents starting from the initial position S and applying forces until either the goal position G is reached or the block leaves the field of play (see Figure 1). We abort a trial if a pre-set number of agent actions fail to

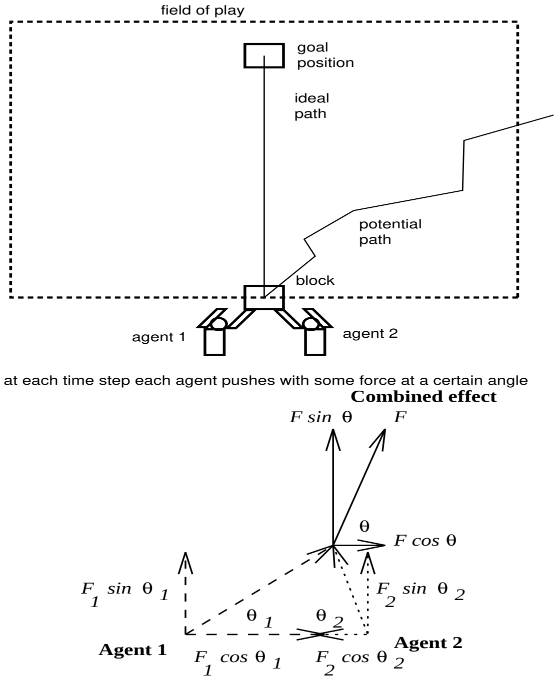

Figure 1: The block pushing problem.

take the block to the goal. This prevents agents from learning policies where they apply no force when the block is resting on the optimal path to the goal but not on the goal itself. The agents are required to learn, through repeated trials, to push the block along the path P to the goal. Although we have used only two agents in our experiments, the solution methodology can be applied without modification to problems with arbitrary number of agents.

Experimental setup

To implement the policy π we chose to use an internal discrete representation for the external continuous space. The force, angle, and the space dimensions were all uniformly discretized. When a particular discrete force or action is selected by the agent, the middle value of the associated continuous range is used as the actual force or angle that is applied on the block.

An experimental run consists of a number of trials during which the system parameters (β, γ, and K) as well as the learning problem (granularity, agent choices) is held constant. The stopping criteria for a run is either that the agents succeed in pushing the block to the goal in N consecutive trials (we have used $N = 10$) or that a maximum number of trials (we have used 1500) have been executed. The latter cases are reported as non-converged runs.

The standard procedure in Q-learning literature of initializing Q values to zero is suitable for most tasks where non-zero feedback is infrequent and hence there is enough opportunity to explore all the actions. Because a non-zero feedback is received after every action in our problem, we found that agents would follow, for an entire run, the path they take in the first trial. This is because they start each trial at the same state, and the only non-zero Q-value for that state is for the action that was chosen at the start trial. Similar reasoning holds for all the other actions chosen in the trial. A possible fix is to choose a fraction of the actions by random choice, or to use a probability distribution over the Q-values to choose actions stochastically. These options, however, lead to very slow convergence. Instead, we chose to initialize the Q-values to a large positive number. This enforced an exploration of the available action options while allowing for convergence after a reasonable number of trials.

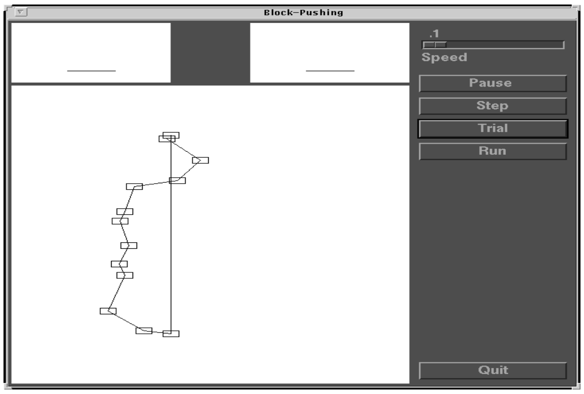

Figure 2: The X/Motif interface for experimentation.

The primary metric for performance evaluation is the average number of trials taken by the system to converge. Information about acquisition of coordination knowledge is obtained by plotting, for different trials, the average distance of the actual path followed from the desired path. Data for all plots and tables in this paper have been averaged over 100 runs.

We have developed a X/Motif interface (see Figure 2) to visualize and control the experiments. It displays the desired path, as well as the current path along which the block is being pushed. The interface allows us to step through trials, run one trial at a time, pause anywhere in the middle of a run, "play" the run at various speeds, and monitor the development of the policy matrices of the agents. By clicking anywhere on the field of play we can see the current best action choice for each agent corresponding to that position.

Choice of system parameters

If the agents learn to push the block along the desired path, the reward that they will receive for the best action choices at each step is equal to the maximum possible value of K. The steady-state values for the Q-values (Q_{ss}) corresponding to optimal action choices can be calculated from the equation:

$$Q_{ss} = (1 - \beta)\, Q_{ss} + \beta\, (K + \gamma\, Q_{ss}).$$

Solving for Q_{ss} in this equation yields a value of $\frac{K}{1-\gamma}$. In order for the agents to explore all actions after the Q-values are initialized at S_I, we require that any new Q value be less than S_I. From similar considerations as above we can show that this will be the case if $S_I \geq \frac{K}{1-\gamma}$. In our experiments we fix the maximum reward K at 50, S_I at 100, and γ at 0.9. Unless otherwise mentioned, we have used $\beta = 0.2$, and allowed each agent to vary both the magnitude and angle of the force they apply on the block.

The first problem we used had starting and goal positions at $(40, 0)$ and $(40, 100)$ respectively. During our initial experiments we found that with an even number of discrete intervals chosen for the angle dimension, an agent cannot push along any line parallel to the y-axis. Hence we used an odd number, 11, of discrete intervals for the angle dimension. The number of discrete intervals for the force dimension is chosen to be 10.

On varying the number of discretization intervals for the state space between 10, 15, and 20, we found the corresponding average number of trials to convergence is 784, 793, and 115 respectively with 82%, 83%, and 100% of the respective runs converging within the specified limit of 1200 trials. This suggests that when the state representation gets too coarse, the agents find it very difficult to learn the optimal policy. This is because the less the number of intervals (the coarser the granularity), the more the variations in reward an agent gets after taking the same action at the same state (each discrete state maps into a larger range of continuous space and hence the agents start from and ends up in physically different locations, the latter resulting in different rewards).

Varying learning rate

We experimented by varying the learning rate, β. The resultant average distance of the actual path from the desired path over the course of a run is plotted in Figure 3 for β values 0.4, 0.6, and 0.8.

In case of the straight path between (40,0) and (40,100), the optimal sequence of actions always puts the block on the same x-position. Since the x-dimension is the only dimension used to represent state, the agents update the same Q-value in their policy matrix in successive steps. We now calculate the number of updates required for the Q-value corresponding to this optimal action before it reaches the steady state value. Note that for the system to converge, it is necessary that only the Q-value for the optimal action at $x = 40$ needs to arrive at its steady state value. This is because the block is initially placed at $x = 40$, and so long as the agents choose their optimal action, it never reaches any other x position. So, the number of updates to reach steady state for the Q-value associated with the optimal action at $x = 40$ should be proportional to the number of trials to convergence for a given run.

In the following, let S_t be the Q-value after t updates and S_I be the initial Q-value. Using Equation 1 and the fact that for the optimal action at the starting position, the *reinforcement* received is K and the next state is the same as the current state, we can write,

$$\begin{aligned} S_{t+1} &= (1-\beta)\, S_t + \beta\,(K + \gamma\, S_t) \\ &= (1-\beta\,(1-\gamma))\, S_t + \beta\, K \\ &= A\, S_t + C \end{aligned} \tag{2}$$

where A and B are constants defined to be equal to $1 - \beta * (1-\gamma)$ and $\beta * K$ respectively. Equation 2 is a difference equation which can be solved using $S_0 = S_I$ to obtain

$$S_t = A^{t+1}\, S_I + \frac{C\,(1 - A^{t+1})}{1 - A}.$$

If we define convergence by the criteria that $|S_{t+1} - S_t| < \epsilon$, where ϵ is an arbitrarily small positive number, then the number of updates t required for convergence can be calculated to be the following:

$$\begin{aligned} t &\geq \frac{\log(\epsilon) - \log(S_I\,(1-A) - C)}{log(A)} \\ &= \frac{\log(\epsilon) - \log(\beta) - \log(S_I\,(1-\gamma) - K)}{log(1 - \beta\,(1-\gamma))} \end{aligned} \tag{3}$$

If we keep γ and S_I constant the above expression can be shown to be a decreasing function of β. This is corroborated by our experiments with varying β while holding $\gamma = 0.1$ (see Figure 3). As β increases, the agents take less number of trials to convergence to the optimal set of actions required to follow the desired path. The other plot in Figure 3 presents a comparison of the theoretical and experimental convergence trends. The first curve in the plot represents the function corresponding to the number of updates required to reach steady state value (with $\epsilon = 0$). The second curve represents the average number of trials required for a run to converge, scaled down by a constant factor of 0.06. The actual ratios between the number of trials to convergence and the values of the expression on the right hand side of the inequality 3 for β equal to 0.4, 0.6, and 0.8 are 24.1, 25.6, and 27.5 respectively (the average number of trials are 95.6, 71.7, and 53; values of the above-mentioned expression are 3.97, 2.8, and 1.93). Given the fact that results are averaged over 100 runs, we can claim that our theoretical analysis provides a good estimate of the relative time required for convergence as the learning rate is changed.

Varying agent capabilities

The next set of experiments was designed to demonstrate the effects of agent capabilities on the time required to converge on the optimal set of actions. In the first of the current set of experiments, one of the agents was chosen to be a "dummy"; it did not exert any force at all. The other agent could only change the angle at which it could apply a constant force on the block. In the second experiment, the latter agent was

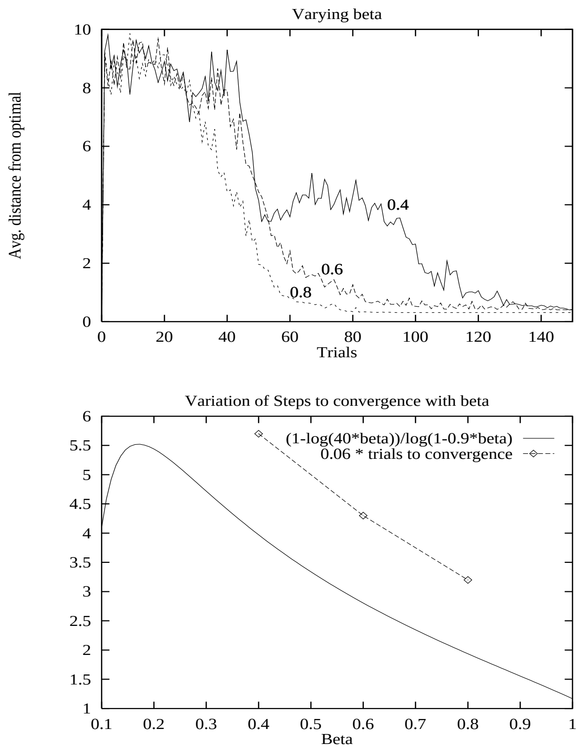

Figure 3: Variation of average distance of actual path from desired path over the course of a run, and the number of updates for convergence of optimal Q-value with changing β ($\gamma = 0.1$, $S_I = 100$).

allowed to vary both force and angle. In the third experiment, both agents were allowed to vary their force and angle. The average number of trials to convergence for the first, second, and third experiment are 55, 431, and 115 respectively. The most interesting result from these experiments is that two agents can learn to coordinate their actions and achieve the desired problem-solving behavior much faster than when a single agent is acting alone. If, however, we simplify the problem of the only active agent by restricting its choice to that of selecting the angle of force, it can learn to solve the problem quickly. If we fix the angle for the only active agent, and allow it to vary only the magnitude of the force, the problem becomes either trivial (if the chosen angle is identical to the angle of the desired path from the starting point) or unsolvable.

Transfer of learning

We designed a set of experiments to demonstrate how learning in one situation can help learning to perform well in a similar situation. The problem with starting and goal locations at (40,0) and (40,100) respectively is used as a reference problem. In addition, we used five other problems with the same starting location and with goal locations at (50,100), (60,100), (70,100),

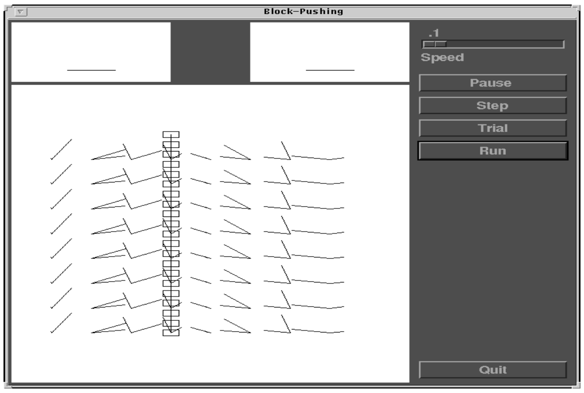

Figure 4: Visualization of agent policy matrices at the end of a successful run.

(80,100), and (90,100) respectively. The corresponding desired paths were obtained by joining the starting and goal locations by straight lines. To demonstrate transfer of learning, we first stored each of the policy matrices that the two agents converged on for the original problem. Next, we ran a set of experiments using each of the new problems, with the agents starting off with their previously stored policy matrices.

We found that there is a linear increase in the number of trials to convergence as the goal in the new problem is placed farther apart from the goal in the initial problem. To determine if this increase was due purely to the distance between the two desired paths, or due to the difficulty in learning to follow certain paths, we ran experiments on the latter problems with agents starting with uniform policies. These experiments reveal that the more the angle between the desired path and the y-axis, the longer the agents take to converge. Learning in the original problem, however, does help in solving these new problems, as evidenced by a $\approx 10\%$ savings in the number of trials to convergence when agents started with the previously learned policy. Using a one-tailed t-test we found that all the differences were significant at the 99% confidence level. This result demonstrates the transfer of learned knowledge between similar problem-solving situations.

Complimentary learning

If the agents were cognizant of the actual constraints and goals of the problem, and knew elementary physics, they could independently calculate the desired action for each of the states that they may enter. The resulting policies would be identical. Our agents, however, have no planning capacity and their knowledge is encoded in the policy matrix. Figure 4 provides a snapshot, at the end of a successfully converged run, of what each agent believes to be its best action choice for each of the possible states in the world. The action choice for each agent at a state is represented by a straight line at the appropriate angle and scaled to represent the magnitude of force. We immediately notice that the individual policies are complimentary rather than being identical. Given a state, the combination of the best actions will bring the block closer to the desired path. In some cases, one of the agents even pushes in the wrong direction while the other agent has to compensate with a larger force to bring the block closer to the desired path. These cases occur in states which are at the edge of the field of play, and have been visited only infrequently. Complementarity of the individual policies, however, are visible for all the states.

Conclusions

In this paper, we have demonstrated that two agents can coordinate to solve a problem better, even without a model for each other, than what they can do alone. We have developed and experimentally verified theoretical predictions of the effects of a particular system parameter, the learning rate, on system convergence. Other experiments show the utility of using knowledge, acquired from learning in one situation, in other similar situations. Additionally, we have demonstrated that agents coordinate by learning complimentary, rather than identical, problem-solving knowledge.

The most surprising result of this paper is that agents can learn coordinated actions without even being aware of each other! This is a clear demonstration of the fact that more complex system behavior can emerge out of relatively simple properties of components of the system. Since agents can learn to coordinate behavior without sharing information, this methodology can be equally applied to both cooperative and non-cooperative domains.

To converge on the optimal policy, agents must repeatedly perform the same task. This aspect of the current approach to agent coordination limits its applicability. Without an appropriate choice of system parameters, the system may take considerable time to converge, or may not converge at all.

In general, agents converged on sub-optimal policies due to incomplete exploration of the state space. We plan to use Boltzmann selection of action in place of deterministic action choice to remedy this problem, though this will lead to slower convergence. We also plan to develop mechanisms to incorporate world models to speed up reinforcement learning as proposed by Sutton (Sutton 1990). We are currently investigating the application of reinforcement learning for resource-sharing problems involving non-benevolent agents.

References

A. B. Barto, R. S. Sutton, and C. Watkins. Sequential decision problems and neural networks. In *Proceedings of 1989 Conference on Neural Information Processing*, 1989.

A. H. Bond and L. Gasser. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann Publishers, San Mateo, CA, 1988.

P. Brazdil, M. Gams, S. Sian, L. Torgo, and W. van de Velde. Learning in distributed systems and multi-agent environments. In *European Working Session on Learning*, Lecture Notes in AI, 482, Berlin, March 1991. Springer Verlag.

P. R. Cohen and C. R. Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3(3):177–212, 1979.

E. H. Durfee and V. R. Lesser. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5), September 1991.

M. S. Fox. An organizational view of distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1):70–80, Jan. 1981.

M. Genesereth, M. Ginsberg, and J. Rosenschein. Cooperation without communications. In *Proceedings of the National Conference on Artificial Intelligence*, pages 51–57, Philadelphia, Pennsylvania, 1986.

J. H. Holland. Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R. Michalski, J. Carbonell, and T. M. Mitchell, editors, *Machine Learning, an artificial intelligence approach: Volume II.* Morgan Kaufman, Los Alamos, CA, 1986.

J. W. Shavlik and T. G. Dietterich. *Readings in Machine Learning.* Morgan Kaufmann, San Mateo, California, 1990.

S. Sian. Adaptation based on cooperative learning in multi-agent systems. In Y. Demazeau and J.-P. Müller, editors, *Decentralize AI*, volume 2, pages 257–272. Elsevier Science Publications, 1991.

R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, Dec. 1980.

R. S. Sutton. Integrated architecture for learning, planning, and reacting based on approximate dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 216–225, 1990.

M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337, June 1993.

C. Watkins. *Learning from Delayed Rewards.* PhD thesis, King's College, Cambridge University, 1989.

G. Weiß. Learning to coordinate actions in multi-agent systems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 311–316, August 1993.