Learning with friends and foes

Mahendra Sekaran
Department of Mathematical & Computer Sciences
University of Tulsa
Tulsa, Oklahoma 74104-3189
mahend@euler.mcs.utulsa.edu

Sandip Sen
Department of Mathematical & Computer Sciences
University of Tulsa
Tulsa, Oklahoma 74104-3189
sandip@kolkata.mcs.utulsa.edu

Abstract

Social agents, both human and computational, inhabiting a world containing multiple active agents, need to coordinate their activities. This is because agents share resources, and without proper coordination or "rules of the road", everybody will be interfering with the plans of others. As such, we need coordination schemes that allow agents to effectively achieve local goals without adversely affecting the problem-solving capabilities of other agents. Researchers in the field of Distributed Artificial Intelligence (DAI) have developed a variety of coordination schemes under different assumptions about agent capabilities and relationships. Whereas some of these research have been motivated by human cognitive biases, others have approached it as an engineering problem of designing the most effective coordination architecture or protocol. We propose reinforcement learning as a coordination mechanism that imposes little cognitive burden on agents. More interestingly, we show that a uniform learning mechanism suffices as a coordination mechanism in both cooperative and adversarial situations. Using an example block-pushing problem domain, we demonstrate that agents can use reinforcement learning algorithms, without explicit information sharing, to develop effective policies to coordinate their actions both with agents acting in unison and with agents acting in opposition.

Introduction

One of the primary goals of artificial intelligence researchers is to develop autonomous agents that are knowledgable and cognizant enough to carry out at least routine activities performed by humans. To be useful in a real-world, however, agents must also inhabit a shared environment, and hence must interact with other agents in the course of their problem-solving activities. Agent interactions may be mutually beneficial or harmful. Beneficial interactions occur when two or more agents can combine their resources and expertise to achieve goals none of them was individually capable of achieving. Harmful interactions occur when the direct or indirect side-effect of a goal achievement or action of one agent makes it impossible or more difficult for one more other agents to achieve their own goals.

A number of coordination schemes have been proposed in DAI literature, using which multiple agents can identify and exploit opportunities for beneficial interactions, and eliminate or restrict harmful interactions. Some of these approaches have been motivated by the way humans argue, negotiate, or influence others through speech and action (Malone, 1987; Cohen & Perrault, 1979; Sycara-Cyranski, 1985). A large majority of DAI approaches to designing coordination schemes, however, have focussed on developing artificial structures (protocols, architectures, conventions, etc.) for efficient problem solving without consideration of human cognitive constraints (Bond & Gasser, 1988). A limitation of the proposed approaches is the fact that they depend critically on the relationship between participating agents. In particular, these coordination schemes rely heavily on information sharing or exchange between agents, the nature of which is largely dependent on whether agents are mutually cooperative or adversarial.

Our proposed approach to coordination involves simultaneous learning by multiple agents working on mutually interacting problems. The particular learning scheme that we have used is known as reinforcement learning, where agents are required to develop policies to map sensations to actions that optimize environmental reward. In contrast with other work on multi-agent learning (Tan, 1993; Weiss, 1993), we do not require that agents exchange or share information with others. In addition to demonstrating that effective coordination knowledge can be induced without explicit or implicit information-sharing, our work provides a coordination scheme that both cooperative and adversarial agents can use without modification. The advantage of such robust coordination scheme is that agents do not have to rely on assumptions about other agents (can I believe the other agents?) or about shared information (has the information been corrupted? is it out-of-date?).

In this paper we use a block pushing domain to illustrate reinforcement learning in both *adversary* and *non-adversary* problems (Nilsson, 1971). The block pushing problems are well-defined (Reitman, 1965) in that the problem components (starting state, goal state, available actions) are completely specified. This domain is, however, *semantically impoverished* rather than being *semantically rich* (Bhaskar & Simon, 1977) (where agents possess and use deep domain knowledge). This is because the focus of our research is more on acquisition of coordination knowledge and less on effective use of prior knowledge.

Reinforcement learning

In reinforcement learning problems (Barto, Sutton, & Watkins, 1989; Holland, 1986; Sutton, 1984; Whitehead & Ballard, 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,

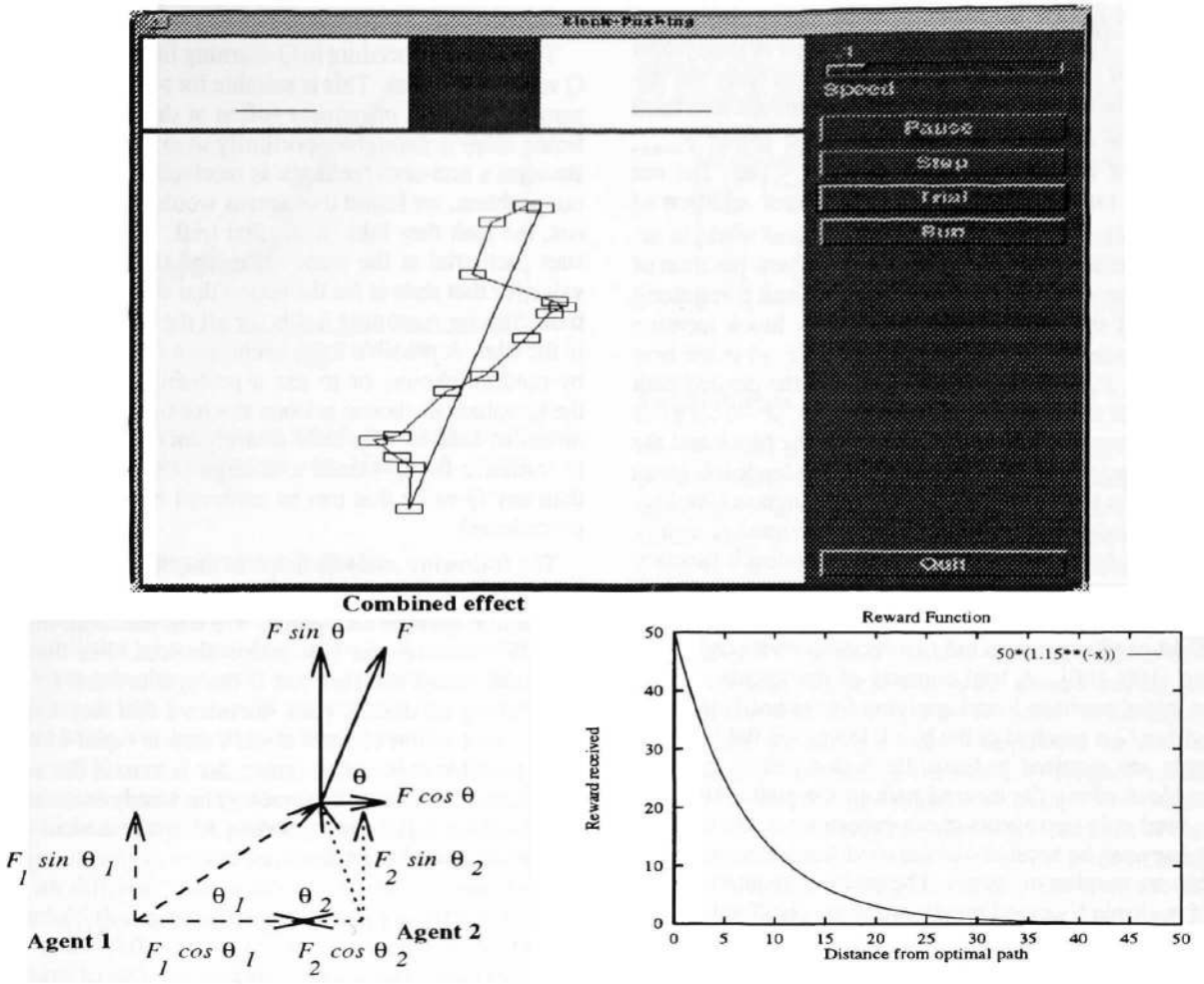

Figure 1: The block pushing problem, reward function, and the X/Motif interface for experimentation.

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 $V_\gamma^\pi \stackrel{def}{=} E\{\sum_{t=0}^{\infty} \gamma^t r_{s,t}^\pi\}$, 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 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:

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

The above update rule is similar to Holland's bucket-brigade (Holland, 1986) and Sutton's temporal-difference (Sutton, 1984) learning scheme.

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 (actually may not even be aware of the presence of the other agent) and yet must choose their actions individually such that the joint task is completed. Agents a_1 and a_2 individually exert forces $\vec{F}_1$ and $\vec{F}_2$ respectively on the object, and the combination of these forces moves

the object. The agents are assumed to be always in contact with the block. 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 $0 \leq \theta \leq \pi$. The net resultant force on the block is found by vector addition of individual forces: $\vec{F} = \vec{F_1} + \vec{F_2}$. The physical world is assumed to be continuous, and 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, $P_{xi}(y)$ is the x-coordinate of the desired path P for agent i for the same y coordinate, $\Delta x = |x - P_{xi}(y)|$ is the distance along the x dimension between the block and the desired path for agent i, then $K * a^{-\Delta x}$ is the feedback given to agent i for its last action. In the reward function (see Figure 1) K is a multiplicative constant (we have used $K = 50$), and $a > 1$ (we have used $a = 1.15$). This feedback function was chosen to transform the actual minimization problem into a maximization problem suitable for Q-learning.

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. The agents are required to learn, through repeated trials, to push the block along the desired path 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. The problem requires the solution of multiple K-armed bandit problems (Holland, 1975), where each of the problems correspond to choosing one of the K possible actions at a state with maximum expected feedback. Actually the problem is harder because the underlying probability distributions for action feedbacks are not known *a priori* (we do not know the means and standard deviations for the feedbacks) and are determined by the evolving policies of both agents, and hence, are dynamically changing (rather than being static, as usually assumed in the K-armed bandit problem). Figure 1 presents a simple pictorial representation of the problem we have described above.

Experimental setup

We now describe some design considerations for implementing the Q-learning procedure for our experiments. 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 (discretizations, agent choices) is held constant. The stopping criteria for a run is that any one of the following three conditions is satisfied: the agents succeed in pushing the block to the goal in N consecutive trials (we have used $N = 10$), difference between agent policy matrices on successive trials is less than ϵ for N successive trials, a maximum number of trials (we have used 1500) have been executed.

The normal procedure in Q-learning literature is to initialize Q values to be zero. This is suitable for most tasks where non-zero feedback is infrequent (often at the end of a trial) 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 (larger than any Q-value that can be achieved by the regular update procedures).

The following analysis helps in the choice of a sufficiently large initial Q-value such that complete exploration of available action options take place. We will calculate the steady-state Q-values for the best action choices. For this, we use Equation 1, and the fact that 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 (since Δx is zero if the actual and the desired paths are the same). 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 0.1. For the experiments in this paper, 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 average number of trials to convergence as the primary metric for evaluating the performance of the system. This value does not, however, provide any information about how agents improve their coordination knowledge over repeated trials. The latter is obtained by plotting, for different trials, the average distance of the actual path followed from the desired path. Results presented in this paper have been averaged over 100 runs.

We have developed a X/Motif interface (see Figure 1) through which we can visualize and control the experiments. The window 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.

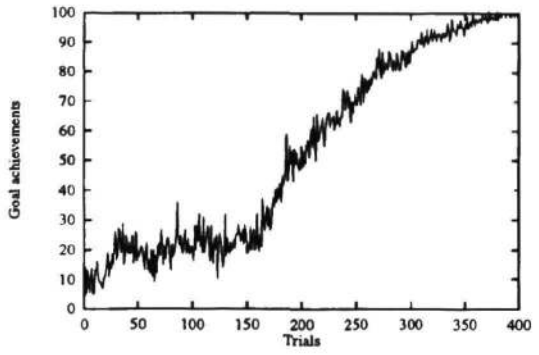

Figure 2: Percentage of runs in which the block reaches the goal on different trials (averaged over 100 runs).

Experiments in cooperative domain

We used a typical problem to evaluate the effects of changing agent capabilities when they are working on a common goal. This problem was chosen with starting position at $(40, 0)$ and goal position at $(60, 100)$, with the straight line between the two points being the desired path (see Figure 1).

We ran 100 runs with different random seeds and found that agents quickly learned to coordinate their actions such that the block is pushed along the optimal path. Figure 2 shows the percentage of runs in which the block was pushed to the goal location on different trial numbers. In only a small number of runs could the agents push the block to the goal on the first few trials. After about 150 trials, the initial exploratory phase seems to be over, and the agents could push the block to the goal with more consistency. By about 400 trials the agents were successful in consistently pushing the block to the goal on all runs. On closer investigation we found that the two agents have learnt complimentary policies. That is, they are not pushing at the same angle; rather, the individual force vectors are such that the resultant force vector lies along the optimal path.

We performed another set of experiments where one agent was a dummy, and the other agent was the sole active agent pushing the block. An interesting observation about this set of experiments is the following: we found that when both agents cooperated to push the block, they converged on the optimal path in less number of trials than when only one agent was pushing (the other agent was a dummy in this case). On examining the policy matrices after convergence, we found that when both agents were operating, less number of policy matrix values differed from their initial setting than when only one agent was pushing the block. This implies that joint action is constraining the block to a region in the state space such that less number of policy values have to be accurately learned. For example, we found that the event of failure to reach the goal in a trial because the block left the playing field on crossing either the x=0 or the x=100 boundaries, was less frequent when two agents were pushing the block compared to when only one agent was pushing the block. Because less number of policy matrix values are to be learned when both agents are operating, runs converge quicker than in the case

Table 1: Trials for agent A to reach its goal acting against agent B trying to reach its own goal (maximum forces applied by agents A and B are F_A and F_B respectively).

F_A	F_B	Trials taken to reach Goal of Agent A
10	1	81
10	2	111
10	3	115
10	4	197
10	5	268

Experiments in adversarial domain

We designed a set of experiments in which two agents are provided different feedback for the same block location. The agents are assigned to push the same block to two different goals along different paths. Hence, the action of each of them adversely affects the goal achievement of the other agent. The maximum force (we refer to this as strength) of one agent was chosen as 10 units, while the maximum force of the other agent was varied. The other variable was the number of discrete action options available within the given force range. When there is considerable disparity between the strengths of the two agents, the stronger agent overpowers the weaker agent, and succeeds in pushing the block to its goal location (see Figure 3). The average number of trials to convergence (see Table 1), however, indicates that as the strength of the weaker agent is increased, the stronger agent finds it increasingly difficult to attain its goal. For these experiments, the strong and the weak agents had respectively 11 (between 0-10) and 2 (0 and its maximum strength) force options to choose from.

When the number of force discretizations for the weak agent is increased from 2 to 10, we find that the stronger agent finds it more difficult to push the block to its own goal. If we increase the maximum force of the weak agent closer to the maximum force of the stronger agent, we find that neither of them is able to push the block to its desired goal. At the of a run, we find that the final converged path lies in between their individual desired paths. As the strength of the weaker agent increases, this path moves away from the desired path of the stronger agent, and ultimately lies midway between their individual desired paths when both agents are equally strong.

Intuitively, an agent should be able to 'overpower' another agent whenever it is stronger. Why is this not happening? The answer lies in the stochastic variability of feedback received for the same action at the same state, and the deterministic choice of the action corresponding to the maximal policy matrix entry. When an agent chooses an action at a state it can receive one of several different feedbacks depending on the action chosen by the other agent. We define the optimal action choice for a state x to be the action A_x that has the highest average feedback F_x. Suppose the first time the agent chooses this action at state x it receives a feedback $F_1 < F_x$. Also, let it receive a feedback $F_2 > F_1$ for a non-optimal action A_y it chooses in the same state x. If these were the only two options available in state x, the agent would choose A_y over A_x next time it is in state x, because the former action corresponds to a higher policy matrix entry. If the steady state value of the policy matrix entry for action A_y in state x is

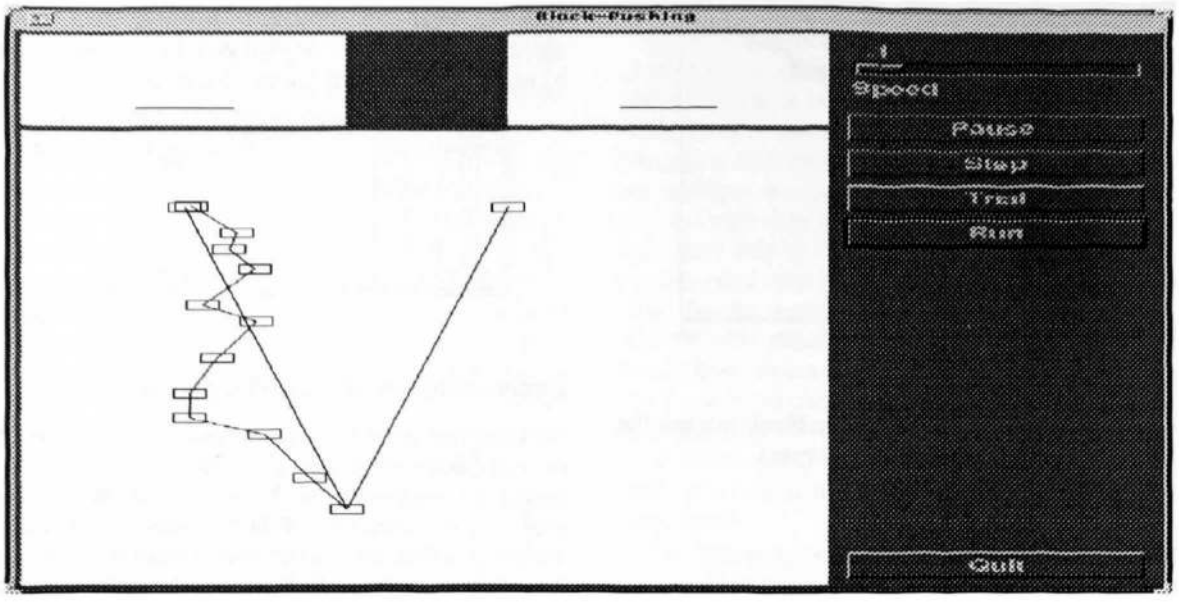

Figure 3: Example trial when agents have conflicting goals.

greater than the policy matrix entry for action A_x obtained after receiving feedback F_1, the latter action will be never tried again, and hence the agent will converge on a non-optimal policy. This is a quintessential example of the *exploration-exploitation* tradeoff (Holland, 1975). Also, this is more likely to happen when the same action can generate more number of distinct feedbacks (the same action for the stronger agent can produce more distinct feedbacks when the discretizations for weaker agent is increased). A simple remedy to this situation will be to choose a proportion of the actions randomly or to choose actions using a probability distribution over the policy matrix values. Each of these options, however, results in an exponential increase of the trials to convergence. Currently we are developing a simulated annealing (Kirkpatrick, Gelatt, & Vecchi, 1983) based procedure which results in a decrease in the proportion of random choices as the policy matrix converges to its steady state.

Conclusions

Using reinforcement learning schemes, we have shown that agents can learn to achieve their goals in both cooperative and adversarial domains. Neither prior knowledge about domain characteristics nor explicit models about capabilities of other agents are required. This provides a novel paradigm for multi-agent systems through which both friends and foes can concurrently acquire coordination knowledge. A drawback of the proposed approach is that it can only be used in domains where agents repeatedly perform similar tasks. We also found that a deterministic choice of agent actions can lead to sub-optimal policies. Our current research effort involves developing stochastic action choice algorithms that converge on better policies without significantly increasing the time taken to converge on these policies. We are also investigating a resource sharing problem domain, in which agents are required to learn to operate a shared system at a load corresponding to its peak efficiency.

References

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

Bhaskar, R. & Simon, H. (1977). Problem solving in semantically rich domains: An example from engineering thermodynamics. *Cognitive Science*, 1:193–215.

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

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

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press.

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

Kirkpatrick, S., Gelatt, C., & Vecchi, M. (1983). Optimization by simulated reannealing. *Science*, 220:671–680.

Malone, T. W. (1987). Modeling coordination in organizations and markets. *Management Science*, 33(10):1317–1332.

Nilsson, N. (1971). *Problem solving methods in artificial intelligence.* New York, NY: McGraw-Hill.

Reitman, W. (1965). *Cognition and Thought.* New York, NY: Wiley.

Sutton, R. S. (1984). *Temporal Credit Assignment in Reinforcement Learning.* Doctoral Dissertation, Amherst: University of Massachusetts.

Sycara-Cyranski, K. (1985). Arguments of persuasion in labour mediation. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 294–296). Los Angeles, California.

Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the*

Tenth International Conference on Machine Learning (pp. 330–337). San Mateo, CA: Morgan Kaufmann Publishers.

Watkins, C. (1989). *Learning from Delayed Rewards*. Doctoral Dissertation, Cambridge: Cambridge University, King's College.

Weiss, G. (1993). Learning to coordinate actions in multi-agent systems. In *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 311–316).

Whitehead, S. D. & Ballard, D. H. (1990). Active perception and reinforcement learning. In *Proceedings of the Seventh International Conference on Machine Learning* (pp. 179–188). San Mateo, CA: Morgan Kaufmann Publishers.