Multiagent coordination with learning classifier systems

Sandip Sen & Mahendra Sekaran
Department of Mathematical & Computer Sciences,
The University of Tulsa
Phone: (918) 631-2985, FAX: (918) 631-3077
e-mail: sandip@kolkata.mcs.utulsa.edu

1 Introduction

Researchers in the field of Distributed Artificial Intelligence (DAI) [Bond and Gasser, 1988] have developed a variety of agent coordination schemes under different assumptions about agent capabilities and relationships. Most of these schemes rely on shared knowledge or authority relationships between agents. These kinds of information may not be available or may be manipulated by malevolent agents. We have used reinforcement learning [Barto *et al.*, 1989] as a coordination mechanism that imposes little cognitive burden on agents and does not suffer from the above-mentioned shortcomings [Sekaran and Sen, 1994; Sen *et al.*, 1994].

In this paper, we evaluate a particular reinforcement learning methodology, a genetic algorithm based machine learning mechanism known as classifier systems [Holland, 1986] for developing action policies to optimize environmental feedback. Action policies that provide 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 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. Experimental results from a couple of multiagent domains show that classifier systems can be more effective than the more widely used Q-learning scheme for multiagent coordination.

2 Coordination of multiple agents

Multiagent systems are a particular type of DAI system, 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.

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, 1988], speech acts [Cohen and 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 *et al.*, 1986] to aid local decision-making.

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. As flexibility and adaptability are key aspects of intelligent and autonomous behavior, we are interested in investigating mechanisms by which agents can acquire and use coordination knowledge through interactions with its environment (that includes other agents) without having to rely on shared information.

In our ongoing research effort to identify such coordination schemes, we compare the performance of classifier systems and the widely used Q-learning algorithm on a resource sharing problem and a robot navigation problem. We show that classifier systems perform competitively with the Q-learning algorithm [Watkins, 1989] to develop effective coordination schemes even when multiple agents are learning concurrently.

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; Weiß, 1993] 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 using reinforcement learning techniques to 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. Our assumptions are similar to that used by Sandholm and Crites [Sandholm and Crites, 1995], and by Sen, Sekaran and Hale [Sen *et al.*, 1994]; the individual goals of agents in the problem domains discussed in this paper are much more loosely coupled than in their problems.

3 Reinforcement learning

In reinforcement learning problems [Barto *et al.*, 1989] 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 develop a policy to maximize rewards accumulated over time. For evaluating the classifier system paradigm for multiagent reinforcement learning, we compare it with 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] algorithm in classifier systems and Sutton's temporal-difference [Sutton, 1984] learning scheme. The similarities of Q-learning and classifier systems have been analyzed in [Dorigo and Bersini, 1994].

Classifier systems are rule based systems that learn by adjusting rule strengths from feedback and by discovering better rules using genetic algorithms. In this paper, we will use simplified classifier systems where all possible message action pairs are explicitly stored and classifiers have one condition and one action. These assumptions are similar to those made by Dorigo and Bersini [Dorigo and Bersini, 1994]; we also use their notation to describe a classifier i by (c_i, a_i), where c_i and a_i are respectively the condition and action parts of the classifier. $S_t(c_i, a_i)$ gives the strength of classifier i at time step t. We first describe how the classifier system performs and then discuss two different feedback distribution schemes, namely the Bucket Brigade algorithm (BBA), and the Profit Sharing Plan (PSP).

All classifiers are initialized to some default strength. At each time step of problem solving, an input message is received from the environment and matched with the classifier rules to form a matchset, $\mathcal{M}$. One of these classifiers is chosen to fire and based on its action, a feedback may be received from the environment. Then the strengths of the classifier rules are adjusted. This cycle is repeated for a given number of time steps. A series of cycles constitute a *trial* of the classifier system. In the BBA scheme, when a classifier is chosen to fire, its strength is increased by the environmental feedback. But before that, a fraction α of its strength is removed and added to the strength of the classifier who fired in the last time cycle. So, if classifier i fires at time step t, produces external feedback of R, and classifier j fires at the next time step, the following equations gives the strength update of classifier i:

$$S_{t+1}(c_i, a_i) = (1-\alpha) * S_t(c_i, a_i) + \alpha * (R + S_{t+1}(c_j, a_j)).$$

We now describe the profit sharing plan (PSP) strength-updating scheme [Grefenstette, 1988] used in classifier systems. In this method, problem solving is divided into episodes in between receipts of external reward. A rule is said to be active in a period if it fired in at least one of the cycles in that episode. At the end of episode e, the strength of each active rule i in that episode is updated as follows:

$$S_{e+1}(c_i, a_i) = S_e(c_i, a_i) + \alpha * (R_e - S_e(c_i, a_i)),$$

where R_e is the external reward received at the end of the episode. We have experimented with two methods of choosing a classifier to fire given the matchset. In the more traditional method, a classifier $i \in \mathcal{M}$ at time t is chosen with a probability given by $\frac{S_t(c_i, a_i)}{\sum_{d \in \mathcal{M}} S_t(c_d, a_d)}$. We call this fitness proportionate PSP or PSP(FP). In the other method of action choice, the classifier with the highest fitness in $\mathcal{M}$ is chosen 90% of the time, and a random classifier from $\mathcal{M}$ is chosen in the rest 10% cases (Mahadevan uses such an action choosing mechanism for Q-learning in [Mahadevan, 1993]). We call this a semi-random PSP or PSP(SR).

For the Q-learning algorithm, we stop a run when the algebraic difference of the policies at the end of neighboring trials is below a threshold for 10 consecutive trials. With this convergence criterion, however, the classifier systems ran too long for us to collect reasonable data. Instead, every 10th trial, we ran the classifier system (both with BBA and PSP) with a deterministic action choice over the entire trial. We stopped a run of the classifier system if the differences of the total environmental feedback received by the system on neighboring deterministic trials were below a small threshold for 10 consecutive deterministic trials.

We study the performance of classifier systems and Q-learning on two different domains. In the resource sharing domain, agents receive feedback only at the end of each trial (delayed), and only one of the agents is learning (the other agent has a fixed load distribution). In the robot navigation domain, agents receive feedback after each step (immediate), and both of the agents learn concurrently.

4 Resource sharing problem

The resource sharing problem assumes two agents sharing a common resource or channel, with each of them trying to distribute their load on the system so as to achieve maximum utility. In our version of the problem, it is assumed that one agent has already applied some load distributed over a fixed time period, and the other agent is learning to distribute its load on the system without any knowledge of the current distribution. A maximum load of L can be applied on the system at any point in time (loads in excess of this do not receive any utility). Related work on adaptive load balancing has been reported in [Schaerf *et al.*, 1995].

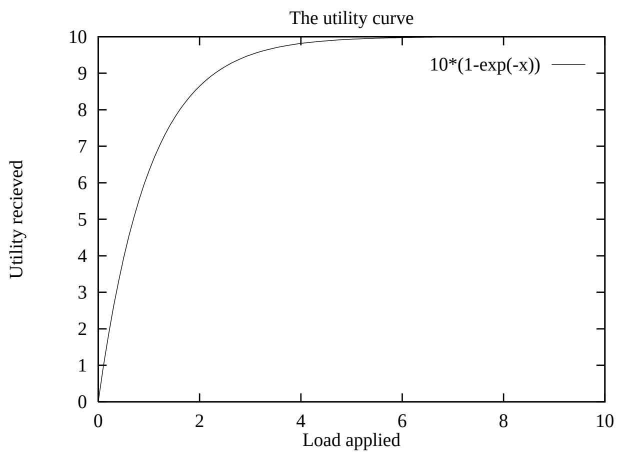

Figure 1: Curve depicting the utility received for a given load

The second agent can use K load-hours of the channel. If it applies a load of k_t load-hours on the system at time step t, when the first agent has applied l_t load-hours, the utility it receives is $u(l_t, k_t) = U(\max(L, k_t + l_t)) - U(\max(L, l_t)$, where U is the utility function in Figure 1, and $L = 10$ is the maximum load allowed on the system. The total feedback it gets at the end of T time steps is $\sum_{t=1}^{T} u(l_t, k_t)$. This problem requires the second agent to distribute its load around the loads imposed by the first agent in order to obtain maximum utility. The problem is that the second agent have no direct information about the load distribution on the system. This is a typical situation in reinforcement learning problems where the agent has to choose its actions based only on scalar feedback.

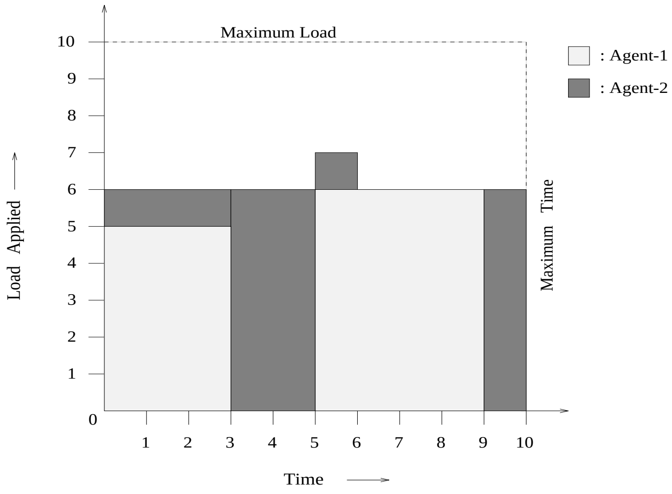

Figure 2: A resource sharing problem.

A single trial consists of an episode of applying loads until T time steps is completed or until the agent has exhausted its load-hours, whichever occurs earlier. Thus, through consecutive such trials the agent learns to distribute its load on the system in an optimal fashion. Figure 2 presents the load distribution used by the first agent as well as one of several optimal load distributions for the second agent in the particular problem we have used for experiments ($T = 10$ in this problem).

Since the resource sharing problem produces rewards only after a series of actions are performed we used the PSP method of payoff distribution with the classifier system. Though BBA can also be used for payoff distribution in this problem, our initial experiments showed that PSP performed much better than BBA on this problem. The parameter values are $\beta = 0.85$, $\gamma = 0.9$ for Q-learning and $\alpha = 0.1$ for PSP.

In this set of experiments the fitness-proportionate PSP did not converge even after 150,000 trials. Experimental results comparing Q-learning and semi-random PSP, PSP(SR), based classifier systems on the resource sharing problem is displayed in Figure 3. Results are averaged over 50 runs of both systems. Though both methods find the optimal load distribution in some of the runs, more often than not they settle for a less than optimal, but reasonably good distribution. PSP takes about twice as long to converge but produces a better load distribution on the average. The difference in performance is found to be significant at the 99% confidence level using a two-sample t-procedure. We believe this happens because all the active rules directly share the feedback at the end of a trial in PSP. In Q-learning, however, the external feedback is passed back to policy elements used early in a trial over successive trials. Interference with different action sequence sharing the same policy element (state-action pair) can produce convergence to sub-optimal solutions.

Typical solutions produced by PSP and Q-learning differed in one important characteristic. PSP solutions will save some of its load for the last empty time-slot, whereas Q-learning solutions use up all the available load before

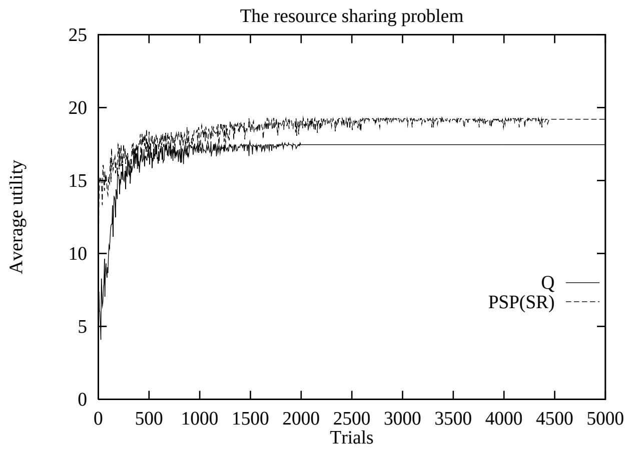

Figure 3: Comparison of PSP and Q-learning on the resource sharing problem.

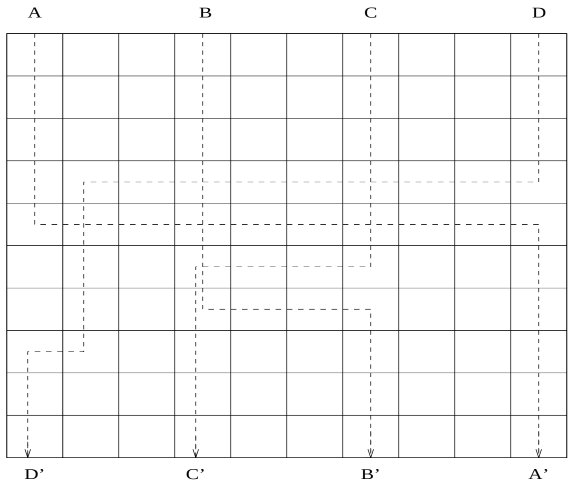

Figure 4: A robot navigation problem.

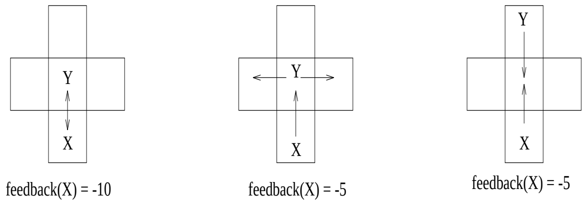

Figure 5: Feedback for agent X when it causes different types of collisions (given the action of the other agent in the collision).

that. Since PSP is able to utilize the last empty time slot on the channel, it produces better utility than Q-learning.

The above results show two things: 1) an agent can effectively use a classifier system to coordinate its actions effectively with no knowledge about the actions of the other agent using the common resource, 2) semi-random action choice mechanism can be a more effective method for classifier systems than the commonly used fitness-proportionate action choice scheme.

5 Robot navigation problem

We designed a problem in which four agents, A, B, C, and D, are to find the optimal path in a grid world, from given starting locations to their respective goals, A', B',C', and D'. The agents traverse their world using one of the five available operators: *north, south, east, west,* or *hold.* Figure 4 depicts potential paths that each of the agents might choose during their learning process. The goal of the agents is to learn moves that quickly take them to their respective goal locations without colliding with other agents.

Each agent receives feedback based on its move: when it makes a move that takes them *towards* their goal, they receive a feedback of 1; when it makes a move that takes them *away* from their goal, they receive a feedback of -1; when it makes a move that results in *no change* of their distance from their goal (*hold*), they receive a feedback of 0; when it makes a move that results in a *collision*, the feedback is computed as depicted in Fig 5. All agents learn at the same time by updating their individual policies.

Since the robot navigation problem produces rewards after each time step, we have used the BBA method of payoff distribution with the classifier system. The system parameters are $\beta = 0.5$, and $\gamma = 0.8$ for Q-learning and $\alpha = 0.1$ for BBA.

Experimental results on the robot navigation domain comparing Q-learning and a classifier system using BBA for payoff distribution is displayed in Figure 3. Plots show the average number of steps taken by the agents to reach their goals. Lower values of this parameter means agents are learning to find more direct paths to their goals without colliding with each other. Results are averaged over 50 runs for both systems. Q-learning takes as much as 5 times longer to converge when compared to BBA. The final number of steps taken by agents using Q-learning is slightly smaller than the number of steps taken by agents using BBA. We believe that if we make the convergence criteria more strict for the BBA, a better solution can be evolved with more computational effort.

The interesting aspect of this experiment is that all the agents were learning simultaneously and hence it was not obvious that they would find good paths. Typical solutions, however, show that agents stop at the right positions to let others pass through. This avoids collisions. The paths do contain small detours, and hence are not optimal.

6 Conclusions

In this paper we have addressed the problem of developing multiagent coordination strategies with minimal domain knowledge and information sharing between agents. We have compared classifier system based methods and Q-learning algorithms, two reinforcement learning paradigms, to investigate a resource sharing and a robot navigation problem. Our experiments show that the classifier based methods perform very competitively with the Q-learning algorithm, and are able to generate

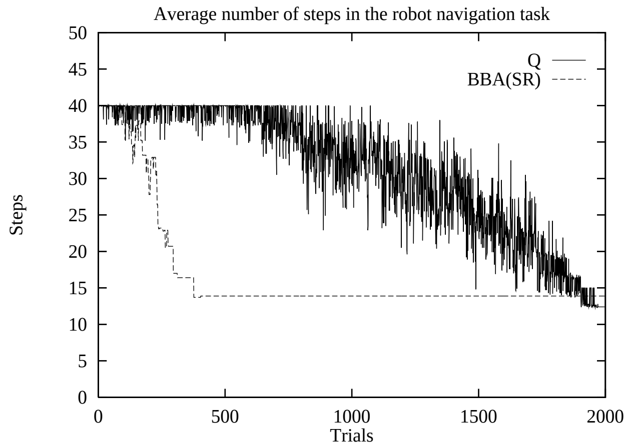

Figure 6: Comparison of BBA and Q-learning on the robot navigation problem.

good solutions to both problems. PSP works well on the resource sharing problem, where an agent is trying to adapt to a fixed strategy used by another agent, and when environmental feedback is received infrequently. Results are particularly encouraging for the robot navigation domain, where all agents are learning simultaneously. A classifier system with the BBA payoff distribution allows agents to coordinate their movements with others without deviating significantly from the optimal path from their start to goal locations.

This paper demonstrates that classifier systems can be used effectively to achieve near-optimal solutions more quickly than Q-learning, as illustrated by the experiments conducted in the robot navigation task. If we enforce a more rigid convergence criteria, classifier systems achieve a better solution than Q-learning through a larger number of trials, as illustrated by the results obtained on the resource sharing domain. We believe however, that either Q-learning or the classifier system can produce better results in a given domain. Identifying the distinguishing features of domains which allow one of these schemes to perform better will be a focus of our future research.

We have also shown that a semi-random choice of actions can be much more productive than the commonly used fitness-proportionate choice of actions with the PSP payoff distribution mechanism. We plan to compare the BBA mechanism with these two methods of payoff distribution.

We would also like to investigate the effects of problem complexity on the number of trials taken for convergence. On the robot navigation domain, for example, we would like to vary both the size of the grid as well as the number of agents moving on the grid to find out the effects on solution quality and convergence time.

Other planned experiments include using world models within classifier systems [Booker, 1988] and combining features of BBA and PSP [Grefenstette, 1988] that would be useful for learning multiagent coordination strategies.

References

[Schaerf *et al.*, 1995] A. Schaerf, Y.Shoham, and M.Tennenholtz. Adaptive load balancing: A study in multiagent learning. *Journal of Artificial Intelligence Research*, 1995. to appear.

[Barto *et al.*, 1989] Andrew B. Barto, Richard S. Sutton, and Chris Watkins. Sequential decision problems and neural networks. In *Proceedings of 1989 Conference on Neural Information Processing*, 1989.

[Bond and Gasser, 1988] Alan H. Bond and Les Gasser. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers, San Mateo, CA, 1988.

[Booker, 1988] L. B. Booker. Classifier systems that learn internal world models. *Machine Learning*, 3:161–192, 1988.

[Brazdil *et al.*, 1991] 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.

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

[Dorigo and Bersini, 1994] Marco Dorigo and Hugues Bersini. A comparison of Q-learning and classifier systems. In *Proceedings of From Animals to Animats, Third International Conference on Simulation of Adaptive Behavior*, 1994.

[Durfee, 1988] Edmund H. Durfee. *Coordination of Distributed Problem Solvers*. Kluwer Academic Publishers, 1988.

[Fox, 1981] Mark S. Fox. An organizational view of distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1):70–80, January 1981. (Also published in *Readings in Distributed Artificial Intelligence*, Alan H. Bond and Les Gasser, editors, pages 140–150, Morgan Kaufmann, 1988.).

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

[Grefenstette, 1988] John Grefenstette. Credit assignment in rule discovery systems. *Machine Learning*, 3(2/3):225–246, 1988.

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

[Mahadevan, 1993] Sridhar Mahadevan. To discount or not to discount in reinforcement learning: A case study comparing R learning and Q learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 205–211, 1993.

[Sekaran and Sen, 1994] Mahendra Sekaran and Sandip Sen. Learning with friends and foes. In *Sixteenth Annual Conference of the Cognitive Science Society*, pages 800–805, 1994.

[Sen *et al.*, 1994] Sandip Sen, Mahendra Sekaran, and John Hale. Learning to coordinate without sharing information. In *National Conference on Artificial Intelligence*, pages 426–431, 1994.

[Sian, 1991] 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.

[Smith, 1980] Reid 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, December 1980.

[Sutton, 1984] Richard S. Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts at Amherst, 1984.

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

[Sandholm and Crites, 1995] T.W. Sandholm and R.H. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, 1995. to appear.

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

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