Advantages of Cooperation Between Reinforcement Learning Agents in Difficult Stochastic Problems

Hamid R. Berenji, David Vengerov
Intelligent Inference Systems Corp.
Computational Sciences Division, MS: 269-2
NASA Ames Research Center
Mountain View, CA 94035
berenji@ptolemy.arc.nasa.gov
vengerov@stanford.edu

Abstract— **This paper presents the first results in understanding the reasons for cooperative advantage between reinforcement learning agents. We consider a cooperation method which consists of using and updating a common policy. We tested this method on a complex fuzzy reinforcement learning problem and found that cooperation brings larger than expected benefits. More precisely, we found that K cooperative agents each learning for N time steps outperform K independent agents each learning in a separate world for K*N time steps. In this paper we explain the observed phenomenon and determine the necessary conditions for its presence in a wide class of reinforcement learning problems.**

I. Introduction

In recent years there has been a considerable amount of interest in multi-agent systems [e.g., Luck, 1997; Sycara, 1998]. One approach to modeling multi-agent learning is to augment the state of each agent with the information about other agents [Littman, 1994; Mataric, 1997; Stone and Veloso, 1999]. However, this approach is difficult to implement in noisy environments where agents cannot reliably discern states and actions of other agents. A more decentralized approach is to give each agent the capability of independent learning in the environment while allowing agents to share learning experience when they find it beneficial.

Several cooperative learning models of this type have been proposed. Kelly and Keating, [1998] considered situations where several robotic agents were learning to avoid stationary and moving obstacles. A common set of fuzzy automata was used by cooperating agents to represent their possible states. Learning consisted of agents taking turns in updating probabilities with which actions were selected in each automaton. In another related work, Tan [1993] simulated several hunters learning to capture a prey in a tile world using reinforcement learning (RL). He studied two RL-based cooperation methods: agents updating a common policy and agents averaging periodically their individual policies. He found that both methods performed equally well on this task, outperforming uncooperative agents that were learning for the same number of time steps. Whitehead [1991] obtained a theoretical upper bound on the increase in the learning rate due to cooperation between multiple agents in a certain class of Markov Decision Processes (MDP's). The considered problem formulation required a finite state space and a deterministic environment with only one goal state. Whitehead showed that under such a formulation, K cooperative agents can learn the optimal policy at most K times faster than a single agent.

In our previous work [Berenji and Vengerov, 1999], we created a complex tile world for investigating the benefits of cooperative reinforcement learning in more general problems than the ones considered previously. In particular, our setup consisted of a partially observable environment with a continuous state space and multiple goals with stochastic properties, which we called "opportunities". We found that for some parameter choices, the performance improvement due to cooperation surpassed the theoretical bound obtained by Whitehead, but for other parameter choices it did not. In this paper we present a detailed analysis of the above phenomenon and describe the necessary conditions for the extra performance improvement. We will start by demonstrating the strongest version of this phenomenon on a simple analytically tractable problem. Then, we will show that a weaker version of this phenomenon is still present in a broad class of more complex problems that are not hand crafted for its existence.

II. One-Dimensional World

Consider the following problem. All events are happening on a line, which, without loss of generality, we can define to be the x-axis. Agent is always placed at 0 at the beginning of every episode. Opportunity 1 is at -1, while opportunity 2 is at +2. The reward of each opportunity is 50 and the step cost is either 60 or -10 with probability (w.p.) 0.5. The agent has two actions: move left (L) and move right (R). If an agent steps into a goal state containing an opportunity, then the episode ends and the agent receives the opportunity reward minus the total path cost since the beginning of the episode.

In the above problem formulation, the agent has only two distinct states, corresponding to its location at 0 and at 1. Call these states x_0 and x_1. There are four Q-values associated with these states: $Q(x_0, L), Q(x_0, R), Q(x_1, L)$, and $Q(x_1, R)$. These Q-values are initialized to 0 at the beginning of every simulation. At every time step, the agent chooses the action that gives the highest Q-value in its current state. The first action in every simulation is to move left. If the two Q-values are equal at any time in the future, the agent chooses the same action as was taken in the previous time step. Agent uses undiscounted Monte Carlo learning to update its Q-values. If the episode began at time T_0 and ended at time T_1, then for $T_0 \leq t < T_1$, the agent performs the following update:

$$Q(x_t, a_t) = Q(x_t, a_t) + \alpha(R_{T_1} + \sum_{\tau=t}^{T_1-1} g(\tau) - Q(x_t, a_t)), \quad (1)$$

where R_{T_1} is the opportunity reward and $g(\tau)$ is the step cost at time τ.

An agent in the above problem will oscillate indefinitely between the two states x_0 and x_1 if it ever happens that $Q(x_0, L) < Q(x_0, R)$ while $Q(x_1, L) > Q(x_1, R)$. Let us call this ordering of Q-values a "failure." It can be verified that at least three time steps are required to generate a failure. More specifically, a failure will occur if the agent incurs the cost of 60 upon the first step to the left, and then incurs costs of -10 and 60 upon the next two steps to the right. If the learning rate $\alpha = 0.1$, then at the end of the first episode the agent will have $Q(x_0, L) = -1$ while $Q(x_1, L) = 0$, and at the end of the second episode the agent will have $Q(x_0, R) = 0$ while $Q(x_1, R) = -1$.

Now consider two agents, each of which is faced with the above problem and which are using and updating the same set of Q-values. It can be verified that at least three time steps are again required to generate a failure, which will happen if both agents encounter the failing sequence of step costs given above. Since the probability of this sequence occurring is 1/8, we see that a single agent fails at the end of 6 time steps w.p. of at least 1/8, while two cooperative agents fail at the end of 3 time steps w.p. of 1/64. Hence, if these agents are faced with a test problem at the end of their learning, then cooperative agents are expected to perform better than the independent agent.

Since we are ultimately interested in continuous state spaces that require value function approximation to be used, we can extend the above conclusions to the following scenario. We assume that opportunity 1 is located at M, and opportunity 2 is located at $N > M$, while the agent can be placed anywhere between them at the beginning of every episode. In this case, the agent can speed up its learning significantly and decrease its memory requirements by generalizing the learned Q-values across states in the following way. The agent still stores only four Q-values, corresponding to two general states: S(small) and L(large). Then the Q-value of an action a (L or R) in a state x (the one positioned at x along the line of the world) is determined as:

$$Q(x, a) = \phi_0(x)Q(S, a) + \phi_1(x)Q(L, a), \quad (2)$$

where $\phi_i(x)$ is monotonically increasing with $\phi_i(x) \geq 0$. This method of Q-value generalization corresponds to fuzzy state generalization, which will be discussed in the next section. It can also be viewed as a special case of value function approximation using a linear combination of features [Tsitsiklis and Van Roy, 1996].

We have applied the above function approximation approach to the original two-state learning problem for the special case when $\phi_0(x) + \phi_1(x) = 1$ and $\phi_0(x_0) = \phi_1(x_1)$. In this case we have $\phi_1(x_0) = \phi_0(x_1) = \epsilon$ and $\phi_0(x_0) = \phi_1(x_1) = 1 - \epsilon$. We found that for any $\epsilon > 0.5$, all previous conclusions still hold. That is, 60, -10, 60 is still the only sequence of step costs resulting in a failure, and the failure probabilities for independent and cooperative agents do not change.

III. Learning Algorithm

The value function approximation approach utilized in the one-dimensional world is a special case of the following more general approach. The Q-values are generalized across states by using a function approximation architecture $Q(x, a, r)$ for approximating $Q(x, a)$, where r is the set of all learned parameters arranged in a single vector. The basic parameter updating rule used by discounted Q-learning or Monte Carlo learning for such an architecture is [Bertsekas and Tsitsiklis, 1996]:

$$r_t \leftarrow r_t + \alpha\delta_t \nabla_{r_t} Q(x_t, a, r_t), \quad (3)$$

where α is the learning rate and δ_t is the Bellman error used in the corresponding learning rule for the look-up table case:

$$Q(x_t, a) \leftarrow Q(x_t, a) + \alpha\delta_t. \quad (4)$$

For example, in the look-up table version of discounted Monte Carlo learning,

$$\delta_t = R_T + \sum_{\tau=t}^{T-1} \gamma^{\tau-t} g(\tau) - Q(x_t, a), \tag{5}$$

where $g(t)$ is the cost incurred at time t, γ is the discounting factor and the summation extends until the end of the episode. In the look-up table version of discounted Q-learning,

$$\delta_t = g(t) + \gamma max_a Q(x_{t+1}, a) - Q(x_t, a). \tag{6}$$

In the general version of discounted Q(λ)-learning, equation (3) becomes:

$$r_t \leftarrow r_t + \delta_t \sum_{\tau=T_0}^{t} (\alpha\lambda)^{t-\tau} \nabla_{r_t} Q(x_t, a, r_t), \tag{7}$$

where T_0 is the time when the current episode began and δ_t is given by equations (5) or (6).

The fuzzy state aggregation given in the one-dimensional world by equation (2) can be generalized as:

$$Q(x, a) = \sum_{k=0}^{K} q(k, a) \mu_k(x), \tag{8}$$

where $q(k, a)$ is the Q-value of taking the action a in the k-th fuzzy state s_k and $\mu_k(x)$ is the degree of membership of state x to s_k. If the action space is continuous, then equation (8) still applies after changing $\mu_k(x)$ to $\mu_k(x, a)$.

With $Q(x, a)$ given by equation (8), $\nabla_{r_t} Q(x_t, a, r_t)$ becomes $\mu_k(x_t)$. Thus, equations (3) and (7) can now be rewritten as matrix equations with each component given by:

$$q(k, a) \leftarrow q(k, a) + \alpha \delta_t \mu_k(x_t). \tag{9}$$

$$q(k, a) \leftarrow q(k, a) + \alpha \delta_t \sum_{\tau=T_0}^{t} (\alpha\lambda)^{t-\tau} \mu_k(x_t). \tag{10}$$

The above equations have a natural interpretation in the realm of fuzzy state aggregations: the Q-value of a fuzzy state-action pair (s_k, a) gets updated proportionally to its contribution to the Q-value of the state-action pair (x_t, a) in equation (8).

If the average cost formulation is used instead of the discounted cost formulation, then equations (9) and (10) still hold, except that δ_t in these equations is given by [Sutton and Barto, 1998]:

$$\delta_t = \sum_{\tau=0}^{T} \gamma^{\tau-t} g(\tau) - Q(x_t, a) - \rho_t \tag{11}$$

for Monte Carlo learning, and by

$$\delta_t = g(t) + \gamma max_a Q(x_{t+1}, a) - Q(x_t, a) - \rho_t \tag{12}$$

for Q(λ)-learning. The quantity ρ represents the average reward per time step of the policy learned so far, to which the average reward from every state-action pair is compared. The quantity ρ is updated at every iteration according to

$$\rho_t \leftarrow \rho_t + \alpha \delta_t. \tag{13}$$

In order to avoid misleading updates, we specify that an agent learning with Q(λ) does not update its Q-values until it reaches the first opportunity.

IV. Two-Dimensional World

We are now ready to apply the insights obtained from the simple one-dimensional world to a more complex two-dimensional world with a continuous state space, as the one considered in our previous work. The world we have constructed is a variation of the Tileworld [Pollack and Ringuette, 1990]. A location in this world is given by a lattice point - a point with integer coordinates. The world evolves in discrete time steps. At each time step, an agent chooses to move either straight or diagonally from its current location to one of the 8 adjacent locations.

Each step in the world has a certain cost, which is calculated using the potential field method: the cost of moving to any location is equal to the potential at the destination. The potential function is generated by randomly choosing locations of deformations of varying strength that generate a potential field. The centers of deformations have real-valued coordinates and are not restricted to lie on the lattice points. The potential at any location due to a certain deformation is given by

$$P = \begin{cases} \frac{h}{exp(\frac{d-0.5}{k})} & \text{if } d > 0.5 \\ h & \text{otherwise,} \end{cases}$$

where h is the height of the deformation, d is the distance to it, and k is a constant specifying the decay rate for the influence of this deformation. Potentials due to all deformations add up to give the final potential at any location. An agent then multiplies this sum by a step cost scaling (SCS) parameter to obtain the cost of moving to the considered location. A deformation is relocated at each time step with a small probability, and its value changes randomly during the relocation. Hence, agents are travelling through a constantly changing potential surface and do not specialize their policies to a particular pattern of deformations.

Opportunities that promise some rewards appear randomly in different locations. Each opportunity has a certain lifetime of M, which specifies that the opportunity

can expire at any time step with probability $\frac{1}{M}$. The reward and the mean lifetime of every newly appearing opportunity are drawn from specified probability distributions. If an agent moves to the location of an unexpired opportunity, it receives the reward promised by that opportunity minus the total path cost since the beginning of the episode. An opportunity disappears once it has been reached by an agent and a new opportunity appears in a randomly chosen location. After that, a new episode begins. If the opportunity toward which the agent took its last step expires, the agent obtains a negative reward equal to its path cost traveled so far and a new episode begins. The objective of an agent is to maximize the total reward minus the total cost received during the simulation.

Our tile world was designed to reflect the complex tradeoffs that humans encounter in many decision situations. The notion of "opportunity" used in our domain description is equivalent to the notion of "alternative" in decision analysis. One of the main difficulties in applying decision analysis to complex situations is that of putting possible outcomes in the order of preference. In the world described above, the agent has to balance the reward of an opportunity against its mean lifetime to come up with some measure of reward the agent expects to receive. Then, the agent has to balance distance to this opportunity with roughness of the path toward it to come up with some measure of cost the agent expects to incur. Finally, the agent has to balance expected reward with expected cost to come up with the total desirability of the opportunity.

A. Decision-Making Process

In order to choose the direction of motion, the agent considers in turn all existing opportunities. For each one, the agent computes the following four variables:

1. distance to the opportunity
2. reward of the opportunity
3. path roughness to the opportunity
4. mean lifetime of the opportunity

Path roughness to the opportunity gives the agent an approximate measure of the average cost per step on the way to that opportunity. It is obtained by first constructing an ellipse with the major axis extending from the agent's current location to the location of the opportunity and passing some number of units beyond the opportunity. The path roughness is then calculated as the sum of the values of all deformations in that ellipse divided by the area of the ellipse.

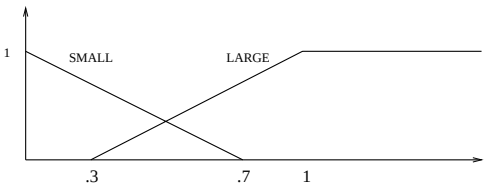

Figure 1: Fuzzy labels used by the agents

The function $\mu_k(x)$ in equation (8) is obtained as follows. The value of each state variable is described using two fuzzy labels: SMALL (S) and LARGE (L). The shapes of these labels are shown in Figure 1. The values of variables with finite ranges are scaled to the range [0,1], while the value of path roughness is scaled so that the expected roughness would correspond to 0.5. The expected roughness is calculated as the expected value of each deformation multiplied by the number of deformations in the tile world and divided by the area of the world. The degree to which an agent belongs to a certain fuzzy state is the minimum of the degrees to which all state variables belong to their corresponding labels in this fuzzy state. Since there are 4 state variables, the total number of possible fuzzy states is 16. Each of these states has a single action A associated with it: move toward the considered opportunity. This gives 16 Q-values that the agent has to learn. More specifically, equation (8) now becomes:

$$\begin{aligned} Q(x,A) &= \mu_0(x)Q(S,S,S,S,A) + \mu_1(x)Q(S,S,S,L,A) \\ &\quad +\ldots+\mu_{15}(x)Q(L,L,L,L,A). \end{aligned} \tag{14}$$

Since the domain has a high degree of stochasticity, no exploration is conducted and the agent takes a step toward the opportunity with the highest Q-value. If the Q-values of several opportunities are equal, the agent chooses among them at random.

B. Preliminary Analysis

A generalization of a "failure" in the one-dimensional world is an ordering of Q-values such that for some X_1, X_2, X_3 we have $Q(S, X_1, X_2, X_3) < Q(L, X_1, X_2, X_3)$, where each X_i is either S or L. This ordering implies that keeping everything else equal, the agent will incorrectly prefer more distant opportunities rather than the closer ones.

The notion of "failure" can be further extended to cover other state variables. That is, keeping everything else equal, the agent should prefer to go for the opportunity with a larger mean lifetime, smaller path roughness, and larger reward. However, the agent can learn incorrectly the reverse of this behavior if at the beginning of the simulation the agent will experience a very large path cost while going toward the opportunity with a large reward,

small path roughness, or large mean lifetime. We will call the orderings of Q-values that lead to such behaviors "incorrect."

An incorrect ordering of Q-values does not have to specify an improper policy, unless all eight pairs of Q-values with respect to the distance variable are inverted. Also, unlike in the one-dimensional world, such an ordering can be unlearned using MC learning. For example, if the agent always goes for the most distant opportunity, then eventually such an opportunity will expire punishing the agent with the incurred path cost.

The intuition obtained from the one-dimensional world can be used to show that individual agent will learn an incorrect ordering with a higher probability than two cooperative agents. As an example, consider the case of incorrect preference of distance. Let's say that $dQ = Q(S, X_1, X_2, X_3) - Q(L, X_1, X_2, X_3)$ follows a random walk during learning with fixed increments of d and that $d < 0$ w.p. p and $d > 0$ w.p. $1-p$, where $p < 0.5$. Initially, $dQ = 0$. After the first episode, a single agent will learn an incorrect ordering of $dQ < 0$ w.p. p. If two cooperative agents end their episodes simultaneously, they will learn an incorrect ordering w.p. $p^2 < p$. Hence, we would expect cooperative agents to perform better when there is a chance of learning an incorrect ordering of Q-values. The above reasoning will apply if at any later point in learning dQ will come close enough to 0 that an episode with a high path cost can make it negative. In the next section, we describe the experimental setup for testing the above conclusions.

C. Experimental Setup

In order to demonstrate clearly the cooperative advantage across different learning algorithms, we use a learning problem that is more difficult than the one considered in our previous work [Berenji and Vengerov, 1999]. More precisely, we choose parameters giving higher probability of learning an incorrect ordering of Q-values.

We used a 11-by-11 tile world in all experiments. There were always 10 opportunities in the world, and whenever one of them would be reached by an agent or would expire, a new one would appear in a random location. The mean lifetime of each appearing opportunity was uniformly distributed between 5 and 20, and its value was uniformly distributed between 30 and 50. The values of appearing deformations were equal to 90 and the decay constant for their influence was 0.2. Each deformation would get relocated with probability 0.1. There were 10 deformations in the world. The step cost scaling parameter was equal to 1.

The learning rate for all algorithms was set to 0.1 in order to provide a natural comparison between simulations of different time spans. The extra cooperative advantage was also observed in our previous work for a decreasing learning rate. We did not consider this case here because even though it does not change the line of reasoning used in this paper, it introduces extra complexity that obscures the studied phenomenon.

cooperative Q-learning	$f = 0.17$	$\mu = 1.92$	$\sigma = 0.04$
independent Q-learning	$f = 0.21$	$\mu = 1.70$	$\sigma = 0.04$
cooperative Q(0.5)-learning	$f = 0.19$	$\mu = 1.10$	$\sigma = 0.05$
independent Q(0.5)-learning	$f = 0.23$	$\mu = 0.85$	$\sigma = 0.05$
cooperative Monte Carlo	$f = 0.13$	$\mu = 1.12$	$\sigma = 0.03$
independent Monte Carlo	$f = 0.15$	$\mu = 0.92$	$\sigma = 0.03$

Table 1: Comparison of cooperative and independent learning for discounted formulation

Cooperation algorithms have the greatest influence on performance at the early stages of learning when agents have not finished exploring sufficiently the whole state space. Therefore, we used a very short time span of fifty or fewer time steps to compare performances of different algorithms.

The performance of an agent is computed as the sum of all rewards minus the sum of all step costs divided by the number of time steps. Performance of the multi-agent team was defined as the average of this measure over all agents. Note that this measure is equivalent to the total reward obtained during the whole simulation.

All performance results represent averages of 20000 simulations. Such a large number of simulations was needed because of a highly stochastic nature of our testing domain. In each simulation, the training period lasted for a specified number of time steps, and the testing period lasted for 100 time steps.

D. Experimental Results and Discussion

This section presents experimental results for some of the most common reinforcement learning algorithms: Q-learning, Monte Carlo learning, and Q(0.5)-learning as a representative of Q(λ)-learning. We used two agents in these experiments, one per world. Cooperative agents were learning for 25 time steps, while independent agents were learning for 50 time steps. Note that performance of two independent agents is just an average of the performance of a single agent over two simulations. As a benchmark, a non-learning agent acting based on initial Q-values obtained performance of -6.95. Such an agent would randomly choose the opportunity to pursue, since all Q-values were initialized to be equal.

Tables 1 and 2 summarize the obtained results. The first column gives the fraction of incorrect Q-value orderings at the end of the learning period. This quantity was

cooperative Q-learning	$f = 0.20$	$\mu = 2.10$	$\sigma = 0.04$
independent Q-learning	$f = 0.25$	$\mu = 1.81$	$\sigma = 0.04$
cooperative Q(0.5)-learning	$f = 0.22$	$\mu = 1.18$	$\sigma = 0.05$
independent Q(0.5)-learning	$f = 0.26$	$\mu = 1.22$	$\sigma = 0.05$
cooperative Monte Carlo	$f = 0.16$	$\mu = 0.92$	$\sigma = 0.05$
independent Monte Carlo	$f = 0.18$	$\mu = 0.59$	$\sigma = 0.05$

Table 2: Comparison of cooperative and independent learning for average cost formulation

computed in each trial by first adding the number of incorrect orderings according to all four state variables. Since each agent used 16 rules, the total number of incorrect orderings of rule pairs according to each variable is 8, and the maximum value of the sum across all four variables is 32. The final fraction was obtained by dividing the resulting sum by 32. The other two columns give performance mean and standard deviation.

As we can see, cooperative agents perform statistically better than independent agents in all considered algorithms except for average cost Q(0.5)-learning. At the same time, the fraction of incorrect orderings of Q-values is always higher for independent agents than for cooperative agents. The sample standard deviation of this fraction was on the order of 0.0003, which makes the reported difference highly significant. This empirical correlation together with the analytical results in the one-dimensional world suggest that the possibility of an incorrect ordering of Q-values is essential to the existence of cooperative advantage.

The case of Q(0.5)-learning suggests that there is a strong counter force to cooperative advantage. One component of this force is the fact that during cooperative learning, agent 1 does not have information about the experiences of agent 2 until the end of its episode. In contrast, during independent learning for a twice longer time interval, information from the previous episode is always available at the current episode. There might be other components to this force hidden in the non-Markovian character of the considered problems.

Agents using Q-learning perform better than those using Q(0.5)-learning. This can be explained by recalling that we allowed agents to learn only for a very short time interval, and TD(λ) methods learn slower for increasing λ [Bertsekas and Tsitsiklis, 1996].

V. Conclusions

In this paper we presented analytical and experimental results shedding some light on the reason why cooperation between reinforcement learning agents can give better results than the ones predicted by Whitehead (1991) for a restricted class of MDP's.

We note that the reasoning used in this paper holds for any number of cooperating agents. Also, this reasoning can be used in a much wider class of problems. For example, any problem where agents move in a space with a distance metric and multiple goals will fall in this category if no function approximation is used or if local feature-based approximations are used. A necessary condition for this is that the probability of learning an "incorrect" ordering is positive.

Acknowledgments: The authors are very grateful to Professor Benjamin Van Roy from the EESOR department of Stanford University for helpful ideas and comments.

References

Hamid R. Berenji, David Vengerov, (1999) "Cooperation and Coordination Between Fuzzy Reinforcement Learning Agents in Continuous State Partially Observable Markov Decision Processes," Proceedings of the 8th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '99), pp. 621-627.

Bertsekas, D. P. and Tsitsiklis, J. N., 1996. Neuro-Dynamic Programming, Athena Scientific.

Kelly, I. D. and Keating, D. A., 1998. "Increased Learning Rates Through the Sharing of Experiences of Multiple Autonomous Mobile Robot Agents," Proceedings of the Seventh IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '98).

Littman, M. L., 1994. "Markov Games as a Framework for Multi-agent Reinforcement Learning," Proceedings of the Eleventh International Conference on Machine Learning, pp. 157-163.

Luck, M., 1997. "Foundations of Multi-Agent Systems: Issues and Directions," The Knowledge Engineering Review, 12(3): 307-308.

Mataric, M. J., 1997. "Reinforcement Learning in the Multi-Robot Domain," Autonomous Robots, 4(1).

Pollack, M. E. and Ringuette, M., 1990. "Introducing the Tileworld: Experimentally Evaluating Agent Architectures," Proceedings of the 8th National Conference on Artificial Intelligence (AAAI '90).

Stone, P. and Veloso, M., 1999. "Team-Partitioned, Opaque-Transition Reinforcement Learning," Proceedings of Third International Conference on Autonomous Agents (Agents '99).

Sutton, R.S., and Barto, A.G., 1998. Reinforcement Learning: An Introduction. MIT Press.

Sycara, K. P. 1998. "Multiagent Systems," AI Magazine, summer 1998, pp. 79-92.

Tan, M. 1993. "Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents," Proceedings of the Tenth International Conference on Machine Learning, pp. 330-337.

Tsitsiklis, J. N. and Van Roy, B. 1996. "Feature-Based Methods for Large-Scale Dynamic Programming," Machine Learning, Vol. 22, pp. 59-64.

Whitehead, S. D., 1991. "A Complexity Analysis of Cooperative Mechanisms in Reinforcement Learning," Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-91), pp. 607-613.