Multiagent Reinforcement Learning for Multi-Robot Systems: A Survey

Multiagent Reinforcement Learning for Multi-Robot Systems: A Survey

Erfu Yang and Dongbing Gu
Department of Computer Science, University of Essex
Wivenhoe Park, Colchester, Essex, CO4 3SQ, United Kingdom
Emails: {eyang, dgu}@essex.ac.uk

***Abstract*— Multiagent reinforcement learning for multi-robot systems is a challenging issue in both robotics and artificial intelligence. With the ever increasing interests in theoretical researches and practical applications, currently there have been a lot of efforts towards providing some solutions to this challenge. However, there are still many difficulties in scaling up the multiagent reinforcement learning to multi-robot systems. The main objective of this paper is to provide a survey, though not completely on the multiagent reinforcement learning in multi-robot systems. After reviewing important advances in this field, some challenging problems and promising research directions are analyzed. A concluding remark is made from the perspectives of the authors.**

I. Introduction

Multi-Robot Systems (MRSs) can often be used to fulfil the tasks that are difficult to be accomplished by an individual robot, especially in the presence of uncertainties, incomplete information, distributed control, and asynchronous computation, etc. The performance of MRSs in redundancy and co-operation contributes to task solutions with a more reliable, faster, or cheaper way. Many practical and potential applications, such as unmanned aerial vehicles (UAVs), spacecraft, autonomous underwater vehicles (AUVs), ground mobile robots, and other robot-based applications in hazardous and/or unknown environments can benefit from the use of MRSs. Therefore, MRSs have received considerable attention during the last decade [1]–[16].

However, there have still been many challenging issues in MRSs. These challenges often involve the realization of basic behaviours , such as trajectory tracking, formation-keeping control, and collision avoidance, or allocating tasks, communication, co-ordinating actions, team reasoning, etc. For a practical multi-robot system, firstly basic behaviours or lower functions must be feasible or available. At the same time upper modules for task allocation and planning have to be designed carefully. When designing MRSs, it is impossible to predict all the potential situation robots may encounter and specify all robot behaviours optimally in advance. Robots in MRSs have to learn from, and adapt to their operating environment and their counterparts. Thus control and learning become two important and challenging problems in MRSs. In this paper we will mainly focus on the learning problems of multi-robot systems and assume that basic behaviours for each participating robot are available.

Currently, there has been a great deal of research on multiagent reinforcement learning (RL) in MRSs [2], [3], [5]–[7], [9]–[11], [15]–[28]. Multiagent reinforcement learning allows participating robots to learn mapping from their states to their actions by rewards or payoffs obtained through interacting with their environment. MRSs can benefit from RL in many aspects. Robots in MRSs are expected to co-ordinate their behaviours to achieve their goals. These robots can either obtain co-operative behaviours or accelerate their learning speed through learning. Among RL algorithms, Q-learning has attracted a great deal of attention [29]–[31]. Explicit presentation of an emergent idea of co-operative behaviours through an individual Q-learning algorithm can be found in [23]. Improving learning efficiency through co-learning was shown by Tan [32]. The study indicates that K co-operative robots learned faster than they did individually. Tan also demonstrated that sharing perception and learning experience can accelerate the learning processes within robot group.

Although Q-learning has been applied to many MRSs, such as forage robots [2], soccer playing robots [5], [23], prey-pursuing robots [32], prey-pursuing robots [17], and moving target observation robots [9], etc., most research work in these applications has only focused on tackling large learning spaces of MRSs. For example, modular Q-learning approaches advocate that a large learning space can be separated into several small learning spaces to ease exploration [17], [23]. Normally, a mediator is needed in these approaches to select optimal policies generated from different modules.

Theoretically, the environment of MRSs are not stationary. Thus the basic assumption for traditional Q-learning working will be violated. Rewards or payoffs learning robots receive depend not only on their own actions but also on the action of other robots. Therefore, the individual Q-learning methods are unable to model the dynamics of simultaneous learners in the shared environment.

Over the last decade there has been increasing interest in extending the individual RL to multiagent systems, particularly MRSs [16], [28], [33]–[50]. From a theoretic viewpoint, this is a very attractive research field since it will expand the range of RL from the realm of simple single-agent to the realm of complex multiagents where there are agents learning simultaneously.

There have been some advances in both multiagent systems and MRSs. The objective of this paper is to review these existing works and analyze some challenging issues from the viewpoint of multiagent RL in MRSs. Moreover, we hope to find some interesting directions for our ongoing research projects.

II. Preliminaries

A. Markov Decision Process

Markov Decision Processes (MDPs) are the mathematical foundation for RL in a single agent environment. Formally, its definition is as follows [31], [42]:

Definition 1 (Markov Decision Process): A Markov Decision Process is a tuple $< \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{R}}, \gamma >$, where $\boldsymbol{\mathcal{S}}$ is a finite discrete set of environment states, $\boldsymbol{\mathcal{A}}$ is a finite discrete set of actions available to the agent, γ $(0 \leq \gamma < 1)$ is a discount factor, $\boldsymbol{\mathcal{T}} : \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{R}} \rightarrow \Pi(\boldsymbol{\mathcal{S}})$ is a transition function giving for each state and action, a probability distribution over states, $\boldsymbol{\mathcal{R}} : \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{R}} \rightarrow \mathbb{R}$ is a reward function of the agent, giving the expected immediate reward received by the agent under each actions in each state.

Definition 2 (Policies): A policy π is denoted for a description of behaviours of an agent. A stationary policy $\pi : \boldsymbol{\mathcal{S}} \rightarrow \Pi(\boldsymbol{\mathcal{A}})$ is a probability distribution over actions to be taken for each state. A deterministic policy is one with probability 1 to some action in each state.

Each MDP has a deterministic stationary optimal policy [42]. In an MDP, the agent acts in a way such as to maximize the long-run value it can expect to gain. Under the discounted objective the factor γ controls how much effect future rewards have on the decisions at each moment. Denoting by $Q^{\pi}(s, a)$ the expected discounted future reward to the agent for starting in a state s and taking an action a for one step then following a policy π, we can define a set of simultaneous linear equations for each state s, i.e., the Q-function for π:

$$Q^{\pi}(s,a) = R(s,a) + \gamma \sum_{s' \in \boldsymbol{\mathcal{S}}} T(s,a,s') \times \sum_{a' \in \boldsymbol{\mathcal{A}}} \pi(s',a') Q^{\pi}(s',a') \quad (1)$$

The Q-function Q^* for the deterministic stationary policy π^* that is optimal for every starting state is defined by a set of equations:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in \boldsymbol{\mathcal{S}}} T(s,a,s') V^*(s) \quad (2)$$

where $V^*(s) = \max_{a' \in \boldsymbol{\mathcal{A}}} Q^*(s',a')$. Thus a greedy policy can be defined according to the Q-function Q^*

Definition 3 (Greedy Policy): A policy π is said to be greedy if it always assigns probability one to an action $\text{argmax}_a Q^*(s,a)$ in state s.

B. Reinforcement Learning

The objective of RL is to learn how to act in a dynamic environment from experience by maximizing some payoff functions or minimizing some cost functions equivalently. In RL, the state dynamics and reinforcement function are at least partially unknown. Thus the learning occurs iteratively and is performed only through trial-and-error methods and reinforcement signals, based on the experience of interactions between the agent and its environment.

C. Q-Learning

Q-learning [29] is a value learning version of RL that learns utility values (Q values) of state and action pairs. It is a form of model-free RL and provides a simple way for agents to learn how to act optimally in controlled Markovian domains. It also can be viewed as a method of asynchronous dynamic programming. In essence Q-learning is a temporal-difference learning method. The objective of Q-learning is to estimate Q values for an optimal policy. During the learning an agent uses its experience to improve its estimate by blending new information into its prior experience. Although there may be more than one optimal policy, the Q^* values are unique [29].

In Q-learning the agent's experience consists of a sequence of distinct episodes. The available experience for an agent in an MDP environment can be described by a sequence of experience tuples $< s_t, a_t, s_t^{'}, r_t >$. Table I shows the scheme of Q-learning.

The individual Q-learning in discrete cases has been proved to converge to optimal values with probability one if state action pairs are visited infinite times and learning rate declines. The following theorem [29] provides a set of conditions under which $Q_t(s,a)$ converges to $Q^*(s,a)$ as $t \to \infty$:

Theorem 1: Given bounded rewards r_t, learning rates $\alpha_t \in [0,1)$, and

$$\sum_{i=1}^{\infty} \alpha_{t^i}(s,a) = \infty, \sum_{i=1}^{\infty} [\alpha_{t^i}(s,a)]^2 < \infty, \forall s,a \quad (4)$$

then $Q_t(s,a) \to Q^*(s,a)$ as $t \to \infty$ with probability one for all s,a. Where $t^i(s,a)$ denotes the index of the i-th time that an action a is tried in a state s.

Although the greedy policy onverges to an optimal policy as $Q_t(s,a) \to Q^*(s,a)$, the agent may not explore a sufficient amount to guarantee the convergent performance if the greedy policy is adopted to choose actions throughout the learning process [51]. To overcome this conflict, a GLIE (Greedy in the Limit with infinite exploration) policy was proposed in [51]. To show the convergence of GLIE policy the following concept is crucial [42]:

Definition 4 (Convergence in Behaviour): An agent converges in behaviour if its action distribution becomes stationary in the limit.

According to Definition 4 Littman [42] pointed out that a GLIE policy need not converge in behaviour since ties in greedy actions are broken arbitrarily. However, an agent with Q-learning will also converge in behaviour if there is a unique

TABLE I

THE Q-LEARNING ALGORITHM

1) Observes the current state s_t.
2) Chooses an action a_t and performs it.
3) Observes the new state $s_t^{'}$ and receives an immediate reward r_t.
4) Adjusts the Q_{t-1} values using a learning factor α_t according to the following rule:

$$Q_t(s,a) = \begin{cases} (1-\alpha_t)Q_{t-1}(x,a) + \alpha_t[r_t + \delta V_{t-1}(s_t^{'})] & \text{if } s = s_t \text{ and } a = a_t \\ Q_{t-1}(x,a) & \text{otherwise} \end{cases} \tag{3}$$

where $V_{t-1}(s^{'}) = \max_{b\in\boldsymbol{\mathcal{A}}} Q_{t-1}(s^{'},b)$

optimal policy and the Q-function converges. The following convergence results come from [42], [51].

Theorem 2: In a single-agent environment, a Q-learning agent will converge to $Q^*(s,a)$ with probability one. Furthermore, an agent following a GLIE policy will be of convergence in behaviour with probability one if the optimal policy is unique.

However, one may find that this theorem is difficult to apply since one often does not know whether there is a unique optimal policy or not. For Q-learning, the situation oriented is usually that the Q^* values rather than one optimal policy are unique [29].

D. Matrix Games

Matrix games are the most elementary type of many players, particularly two-player games [52]. In matrix games players select actions from their available action space and receive rewards that depend on all the other player's actions.

Definition 5 (Matrix Games): A matrix game is given by a tuple $< n,\boldsymbol{\mathcal{A}}_1,\cdots,\boldsymbol{\mathcal{A}}_n, R_1,\cdots,R_n >$, where n is the number of players, $\boldsymbol{\mathcal{A}}_i$ and R_i $(i = 1,\cdots,n)$ are the finite action set and payoff function respectively for player i.

Among matrix games, bimatrix games are often used to formulate the frameworks for multiagent reinforcement learning. Hence we particularly give its definition as follows [53]:

Definition 6 (Bimatrix Games): A bimatrix game is defined by a pair of payoff matrices (M_1, M_2). Each matrix $M_i(i = 1,2)$ has a dimension of $|A_1| \times |A_2|$ and its entry $M_i(a_1,a_2)$ gives the reward of the ith player under a joint action pair $(a_1,a_2), \forall(a_1,a_2) \in\boldsymbol{\mathcal{A}}_1\times \boldsymbol{\mathcal{A}}_2$.

In matrix games the joint actions correspond to particular entries in the payoff matrices. For the reinforcement learning purpose, agents play the same matrix game repeatedly. For applying repeated matrix game theory to multiagent reinforcement learning, the payoff structure must be given explicitly. This requirement seems to be a restriction for applying matrix games to the domains of MRSs where payoff structure is often difficult to define in advance.

E. Stochastic Games

Currently multiagent learning has focused on the theoretic framework of Stochastic Games (SGs) or Markov Games (MGs). SGs extend one-state MG to multi-state cases by modelling state transitions with MDP. Each state in a SG can be viewed as a MG and a SG with one player can be viewed as an MDP. In what follows we first review some important concepts for easing the understanding of the related works in this survey.

Definition 7 (Best-Response): A policy is said to be a best-response to the other player' policies if it is optimal given their policies.

Definition 8 (Nash Equilibrium): A Nash equilibrium is a collection of strategies for each of the players such that each player's strategy is a best-response to the other players' strategies.

At a Nash equilibrium, no player can do better by changing strategies unilaterally given that the other players don't change their Nash strategies. There is at least one Nash equilibrium existed in a game [52], [54].

Definition 9 (Mixed-Strategy Nash Equilibrium): A Mixed-Strategy Nash equilibrium for a bimatrix game is a pair of strategies (π_1^*, π_2^*) for each of the players such that each player's strategy is a best-response to the other player' strategy,mathematically i.e.,

$$\begin{array}{ll} {\pi_1^*}^T M_1 \pi_2^* \geq {\pi_1}^T M_1 \pi_2^* & \forall \pi_1 \in \Pi(\boldsymbol{\mathcal{A}}_1) \\ {\pi_1^*}^T M_2 \pi_2^* \geq {\pi_1^*}^T M_2 \pi_2 & \forall \pi_2 \in \Pi(\boldsymbol{\mathcal{A}}_2) \end{array} \quad (5)$$

where $\Pi(\boldsymbol{\mathcal{A}}_1)$ and $\Pi(\boldsymbol{\mathcal{A}}_2)$ are the probability distributions over the corresponding action spaces of agent 1 and 2, respectively.

F. Game Theory and Multiagent Reinforcement Learning

Game Theory (GT) is explicitly designed for reasoning among multiple players. Players in GT are assumed to act rationally. They always take their best policies to play, rather than play tricks. A great deal of research on multiagent learning has borrowed the theoretic frameworks and notions from SGs [33]–[37], [39], [41]–[45], [47]–[50], [55]–[58]. SGs have been well studied in the field of multiagent reinforcement learning and appear to be a natural and powerful extension of MDPs to multi-agent domains. In the framework of SGs Nash equilibria is an important solution concept for the problem of simultaneously finding optimal policies in the presence of other learning agents. At a Nash equilibrium each agent player is playing optimally with respect to the others under a Nash equilibrium policy. If all the agents are playing a policy at a Nash equilibrium rationally, then no agent could learn a better policy.

III. Theoretic Frameworks for Multiagent Reinforcement Learning

A. SG-Based Frameworks

The framework of SGs (or MGs) is widely adopted by many researchers to model multiagent systems with finite states and actions [33], [42], [50], [59], [60]. Particularly the framework of SGs is exploited in extending Q-learning to multiagent systems. In a SG, all agents select their actions simultaneously. the reward each agent receives depends on the joint actions of all agents and the current state as well as the state transitions according to the Markov property. A reinforcement learning framework of SGs is given by the following formal definition [33], [42], [50], [59].

Definition 10 (Framework of SGs): A learning framework of SGs is described by a tuple $< \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{A}}_1, \cdots, \boldsymbol{\mathcal{A}}_n, T, R_1, \cdots, R_n, \gamma >$, where

- $\boldsymbol{\mathcal{S}}$ is a finite state space;
- $\boldsymbol{\mathcal{A}}_1, \cdots, \boldsymbol{\mathcal{A}}_n$ are the corresponding finite sets of actions available to each agent.
- $T : \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{A}}_1 \times \cdots \times \boldsymbol{\mathcal{A}}_n \rightarrow \Pi(\boldsymbol{\mathcal{S}})$ is a state transition function, given each state and one action from each agent. Here $\Pi(\boldsymbol{\mathcal{S}})$ is a probability distribution over the state space $\boldsymbol{\mathcal{S}}$.
- $R_i : \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{A}}_1 \times \cdots \times \boldsymbol{\mathcal{A}}_n \rightarrow \mathbb{R} (i = 1, \cdots, n)$ represents a reward function for each agent.
- $0 \leq \gamma < 1$ is the discount factor.

In such a learning framework of SGs, learning agents attempt to maximize their expected sum of discounted rewards. Correspondingly a set of Q-functions for agent i $(i = 1, \cdots, n)$ can be defined according to their stationary policies $\pi_1, \cdots, \pi_n$. Unlike a single-agent system, in multiagent systems the joint actions determine the next state and rewards to each agent. After selecting actions, the agents are transitioned to the next state and receive their rewards.

B. *Fictitious Play Framework*

In a known SG, the framework of fictitious play can be used as a technique to finding equilibria. For a learning paradigm, fictitious play can also be applied to form a theoretical framework [60]. It provides a quite simple learning model. In the framework of fictitious play, the algorithm maintains information about the average estimated sum of future discounted rewards. According to the Q-functions of the agents, the fictitious play method deterministically chooses the actions for each agent that would have done the best in the past. For computing the estimated sum of future discounted rewards, a simple temporal difference backup may be used.

Compared with the framework of SGs, the main merit of fictitious play is that it is capable of finding equilibria in both zero-sum games and some classes of general-sum games [60]. One obvious disadvantage of this framework is that fictitious play merely adopts deterministic policies and cannot play stochastic strategies. Hence it is hard to apply in zero-sum games because it can only find an equilibrium policy but does not actually play according to that policy [60]. In addition learning stability is another serious problem. Since the fictitious play framework is of inherent discontinuity, a small change in the data could lead to an abrupt in behaviour [49]. To overcome this unstable problem, many variants of fictitious play have been developed, see [49] as well as the literature therein.

C. *Bayesian Framework*

The multiagent reinforcement learning algorithms developed from the SG framework, such as Minimax-Q, Nash-Q, etc., always require to converge to desirable equilibria. Thus, sufficient exploration of strategy space is needed before convergence can be established. Solutions to multiagent reinforcement learning problems are usually based on equilibrium. Thus to obtain an optimal policy, agents have to find and even identify the equilibria before the policy is used at the current state.

A Bayesian framework for exploration in multi-agent reinforcement learning systems was proposed in [47], [49]. The Bayesian framework is a model-based reinforcement learning model. In this framework the learning agent can use priors to reason about how its action will influence the behaviours of other agents. Thus, some prior density over possible dynamics and reward distribution have to be known by a learning agent in advance.

A basic assumption in the Bayesian framework is that the learning agent is able to observe the actions taken by all agents, the resulting game state, and rewards received by other agents. Of course, this assumption will have no problem for the coordination of multiagents, but it will restrict its applications in other settings where opponent agents generally will not broadcast their information to the others.

To establish the belief, a learning agent under the Bayesian framework has some priors , such as probability distribution over the state space as well as the possible strategy space. The belief is then updated during the learning by observing the results of its actions and action choices of other agents. In order to predict accurately the actions of other agents, the learning agent has to record and maintain appropriate observable history. In [47], [49] it is assumed that the learning agent can keep track of sufficient history to make such predictions.

Besides the aforementioned assumptions, in [47], [49] there are two extra assumptions on the belief. First, the priors over models can be factored into independent local models for both rewards and transitions. Second, it needs to be assumed that the belief about opponent strategies also can be factored and represented in some convenient form [47], [49].

D. Policy Iteration Framework

Unlike the value iteration frameworks, the policy iteration framework can provide a direct way to find the optimal strategy in the policy space. Under the policy iteration framework Bowling and Veloso [61] proposed a WoLF-PHC algorithm if the other agents are assumed to be playing stationary policies. The other works following the thinking lines of [61] can be found in [53], [62]. It seems to have no other reported works on the algorithms under this framework when the other agents in the system are considered to learn simultaneously. Compared with the aforementioned frameworks, many researches for policy iteration reinforcement learning in multiagent systems still need to be done in the future. Fortunately, there already have been many research results on policy iteration algorithms, for instance, one may refer to [63] as well as the literature therein. Thus one possible way is to extend the existing policy iteration algorithms in single-agent systems to the field of multiagent systems.

IV. Multiagent Reinforcement Learning Algorithms

The difference between single-agent and multiagent system exists in the environments. In multiagent systems other adapting agents make the environment no longer stationary, violating the Markov property that traditional single agent behavior learning relies upon.

For individual robot learning, the traditional Q-learning has been successfully applied to many paradigms. Some researchers also apply Q-learning in a straightforward fashion to each agent in a multiagent system. However, the aforementioned fact that the environment is no longer stationary in multiagent system is usually neglected. Over the last decade many researchers have made efforts to use the RL methodology, particularly the Q-learning framework as an alternative approach to the learning of MRSs. As pointed out early, the basic assumption for traditional Q-learning working is violated in the case of MRSs.

A. Minimax-Q Learning Algorithm

Under SG framework, Littman [33] proposed a *Minimax-Q* learning algorithm for zero-sum games in which learning player maximizes its payoffs in the worst situation. The players' interests in the game are opposite. Essentially the Minimax-Q learning is a value-function reinforcement learning algorithm. In the Minimax-Q learning the player always try to maximize its expected value in the face of the worst-possible action choice of the opponent. Hence the player would become more cautious after learning. To calculate the probability distribution or the optimal policy of the player, Littman [33] simply used linear programming.

An illustrating version of the Minimax-Q learning algorithm is shown in Table II. The Minimax-Q learning algorithm was firstly given in [33], which just included empirical results on a simple zero-sum SG game version of soccer. A complete convergence proof was provided in the works thereafter [34], [39], [42], which can be summarized in the following theorem:

Theorem 3: In a two-player zero-sum multiagent SG environment, an agent following the Minimax-Q learning algorithm will converge to the optimal Q-function with probability one. Furthermore, an agent using a GLIE policy will converge in behaviour with probability one if the limit equilibrium is unique.

The Minimax-Q learning algorithm may provide a safe policy in that it can be performed regardless

TABLE II

THE MINIMAX-Q LEARNING ALGORITHM

1) Initialize $V(s), Q(s, a_1, a_2)$ for all $s \in \boldsymbol{\mathcal{S}}, a_1 \in \boldsymbol{\mathcal{A}}_1$, and $a_2 \in \boldsymbol{\mathcal{A}}_2$.
2) Choose an action,
   a) With an exploring probability, return an action uniformly at random.
   b) Otherwise, return action a_1 with probability $\pi(s, a_1)$
3) Learn,
   a) After receiving a reward r for moving from state s to $s^{'}$ via action a_1 and opponent's action a_2,
   b) Update,
   $$Q(s, a_1, a_2) \leftarrow (1-\alpha)Q(s, a_1, a_2) + \alpha(r + \gamma V(s^{'})$$
   c) Using linear programming to find $\pi(s, \cdot)$ such that:
   $$\pi(s, \cdot) \leftarrow \arg \max_{\pi^{'}(s,\cdot)\in\Pi(\boldsymbol{\mathcal{A}}_1)} \min_{a_2^{'}\in\boldsymbol{\mathcal{A}}_2} \Sigma_{a_1^{'}} \pi(s, a_1^{'}) Q(s, a_1^{'}, a_2^{'})$$
   d) Let
   $$V(s) \leftarrow \min_{a_2^{'}\in\boldsymbol{\mathcal{A}}_2} \Sigma_{a_1^{'}} \pi(s, a_1^{'}) Q(s, a_1^{'}, a_2^{'})$$
   e) Let $\alpha \leftarrow \alpha\varepsilon$, where ε is a decaying rate for learning parameter α.

of the existence of its opponent [42]. The policy used in the Minimax-Q learning algorithm can guarantee that it receives the largest value possible in the absence of knowledge of the opponent's policy. Although the Minimax-Q learning algorithm manifest many advantages in the domain of two-player zero-sum multiagent SG environment, an explicit drawback of this algorithm is that it is very slow to learn since in each episode and in each state a linear programming is needed. The use of linear programming significantly increases the computation cost before the system reaches convergence.

B. Nash-Q Learning Algorithm

Hu and Wellman [37], [50] extended the zero-sum game framework of Littman [33] to general-sum games and developed a *Nash-Q* learning algorithm for multiagent reinforcement learning. To extend Q-learning to the multiagent learning domain, the joint actions of participating agents rather than merely individual actions are needed to take into account. Considering this important difference between single-agent and multiagent reinforcement learning, the Nash-Q learning algorithm needs to maintain Q values for both the learner itself and other players. The idea is to find Nash equilibria at each state in order to obtain Nash equilibrium policies for Q value updating.

To apply the Nash-Q learning algorithm, one has to define the Nash Q-value. A Nash Q-value is defined as the expected sum of discounted rewards when all agents follow specified Nash equilibrium strategies from the next period on. The following several definitions directly come from [50] for understanding the Nash-Q learning algorithm.

Definition 11 (Nash Q-function): The Nash Q-function for the ith agent is defined over $(s, a_1, \cdots, a_n)$ as the sum of its current reward plus its future rewards when all agents follow a joint Nash equilibrium strategy, i.e.,

$$Q_i^*(s, a_1, \cdots, a_n) = r_i(s, a_1, \cdots, a_n) + \gamma \Sigma_{s' \in \boldsymbol{\mathcal{S}}} T(s, s', a_1, \cdots, a_n) V_i(s', \pi_1^*, \cdots, \pi_n^*) \tag{6}$$

where $(\pi_1^*, \cdots, \pi_n^*)$ is a joint Nash equilibrium strategy, $r_i(s, a_1, \cdots, a_n)$ is an one-period reward for ith agent in state s under the joint action $(a_1, \cdots, a_n)$, $V_i(s', \pi_1^*, \cdots, \pi_n^*)$ is the total discounted reward over infinite periods starting from state s' given that the agents follow the equilibrium strategies, $T(s, s', a_1, \cdots, a_n)$ is the probability distribution of state transitions under the joint action $(a_1, \cdots, a_n)$.

Definition 12 (Nash Joint Strategy): A joint strategy $(\pi_1, \cdots, \pi_n)$ constitutes a Nash equilibrium for the stage games $(M_1, \cdots, M_n)$ if,

$$\pi_k \pi_{-k} M_k \geq \hat{\pi}_k \pi_{-k} M_k \text{ for all } \pi_k \in \boldsymbol{\mathcal{A}}_k,$$

where $k = 1, \cdots, n$, π_{-k} is the product of strategies for all agents other than k, i.e., $\pi_{-k} = \pi_1 \cdots \pi_{k-1} \cdot \pi_{k+1} \cdots \pi_n$.

With the previous preliminaries, the Nash-Q learning algorithm now can be summarized in Table III. Hu and Wellman used quadratic programming to find Nash equilibrium in the Nash-Q learning algorithm for general-sum games.

Hu and Wellman [37], [50] has shown that the Nash-Q learning algorithm in multi-player environment converges to Nash equilibrium policies with probability one under some conditions and additional assumptions to the payoff structures. More formally, the main results can be summarized in the following theorem [37], [42], [50]:

Theorem 4: In a multiagent SG environment, an agent following the Nash-Q learning algorithm will converge to the optimal Q-function with probability one as long as all Q-functions encountered have coordination equilibria and these are used in the update rule. Furthermore, the agent using a GLIE policy will converge in behaviour with probability one if the limit equilibrium is unique.

To guarantee the convergence, the Nash-Q learning algorithm needs to know that a Nash equilibrium is either unique or has the same value as all others. Littman [42] has argued the applicability of Theorem 4 and pointed out that it is hard to apply since the strict conditions are difficult to verify in advance. To tackle this difficulty, Littman [44] thereafter proposed a so-called Friend-or-Foe Q-learning (FFQ) algorithm, which will be introduced in the following subsection.

C. Friend-or-Foe Q-learning (FFQ) Algorithm

Motivated by the conditions of Theorem 4 on the convergence of Nash-Q learning, Littman [44] developed a Friend-or-Foe Q-learning (FFQ) algorithm for the RL in general-sum SGs. The main idea is that each agent in the system is identified as being either “friend” or “foe”. Thus, the equilibria can be classified as either coordination or adversarial. Compared with the Nash-Q learning, the FFQ-learning can provide a stronger convergence guarantee.

Littman [44] has presented the following results to prove the convergence of the FFQ-learning algorithm:

Theorem 5: Foe-Q learns values for a Nash equilibrium policy if there is an adversarial equilibrium; Friend-Q learns values for a Nash equilibrium policy if the game has a coordination equilibrium. This is true regardless of opponent behaviour.

Theorem 6: Foe-Q learns a Q-function whose corresponding policy will achieve at least the learned values regardless of the policy selected by the opponent.

Although the convergence property of FFQ-learning has been improved over that of Nash-Q learning algorithm, a complete treatment of general-

TABLE III

THE NASH-Q LEARNING ALGORITHM

1) Initialize:
   a) Give the initial state s^0 and let $t \leftarrow 0$.
   b) Take the ith agent as the learning agent.
   c) For all $s \in \boldsymbol{\mathcal{S}}$ and $a_i \in \boldsymbol{\mathcal{A}}_i$, $i = 1, \cdots, n$, $Q_i^t(s, a_1, \cdots, a_n) \leftarrow 0$.
2) Repeat:
   a) Choose action a_i^t.
   b) Observe $r_1^t, \cdots, r_n^t$; $a_1^t, \cdots, a_n^t$, and $s^{t+1} \leftarrow s^{'}$.
   c) Update Q_i^t for $i = 1, \cdots, n$:

$$Q_i^{t+1}(s, a_1, \cdots, a_n) = (1 - \alpha_t) Q_i^t(s, a_1, \cdots, a_n) + \alpha_t[r_i^t + \gamma NashQ_i^t(s^{'})]$$

3) $t \leftarrow t + 1$.

where $\alpha_t \in (0, 1)$ and the $NashQ_i^t(s^{'})$ is defined as:

$$NashQ_i^t(s^{'}) = \pi_1(s^{'}) \cdots \pi_n(s^{'}).Q_i^t(s^{'})$$

sum stochastic games using Friend-or-Foe concepts is still lacking [44].

In comparison to the Nash-Q learning algorithm, the FFQ-learning does not require learning estimates to the Q-functions of opponents. However, the FFQ-learning still require a very strong condition for application, that is the agent must know how many equilibria there are in game and an equilibrium is known either coordinating or adversarial in advance. The FFQ-learning itself does not provide a way to find a Nash equilibrium or identify a Nash equilibrium as being either a coordination or an adversarial one. Like the Nash-Q learning, the FFQ-learning also cannot apply to the system where neither coordination nor adversarial equilibrium exists.

D. rQ-learning Algorithm

Morales [64] developed a so-called rQ-learning algorithm for dealing with large search space problem. In this algorithm a *r-state* and a *r-action* set need to be defined in advance. A *r-state* is defined by a set of first-order relations, such as *goal_in_front*, *team_robot_to_the_left*, *opponent_robot_with_ball*, etc. A *r-action* is described by a set of pre-conditions, a generalized action, and possibly a set of post-conditions. For a *r-action* to be defined properly, the following condition must be satisfied: if a *r-action* is applicable to a particular instance of a *r-state*, then it should be applicable to all the instances of that *r-state*. The rQ-learning algorithm can reduce the size of search space, the process is given in Table IV.

Although the rQ-learning algorithm seems to be useful for dealing with large search space problem, it may be very difficult to define a *r-state* and a *r-action* set properly, particularly in the case with incomplete knowledge on the concerned MRS. Furthermore, in the *r-state* space there is no guarantee that the defined `r-actions` are adequate to find an optimal sequence of primitive actions and sub-optimal policies can be produced [64].

TABLE IV

THE rQ-LEARNING ALGORITHM

1) Initialize $Q(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{A}})$ ($\boldsymbol{\mathcal{S}}$ is the r-state set and $\boldsymbol{\mathcal{A}}$ is r-action set) arbitrarily.
2) Repeat,
   a) Initialize state s.
   b) $\boldsymbol{\mathcal{S}} \leftarrow \mathrm{rels}(s)$ for evaluating the set of relations over state s,
   c) Repeat for each step of episode,
      i) Choose $\boldsymbol{\mathcal{A}}$ from $\boldsymbol{\mathcal{S}}$ using a persistently exciting policy;
      ii) Choose action a from $\boldsymbol{\mathcal{A}}$ randomly;
      iii) Apply action a and observe $r, s^{'}$
      iv) Update

$$\boldsymbol{\mathcal{S}}^{'} \leftarrow \mathrm{rels}(s)$$
$$Q(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{A}}) \leftarrow Q(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{A}}) + \alpha(r + \gamma \max_{\boldsymbol{\mathcal{A}}'} Q(\boldsymbol{\mathcal{S}}^{'}, \boldsymbol{\mathcal{A}}^{'}) - Q(\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{A}}))$$
$$\boldsymbol{\mathcal{S}} \leftarrow \boldsymbol{\mathcal{S}}^{'}$$

   d) until s is terminal.

E. Fictitious Play Algorithm

Since the Nash-equilibrium-based learning has difficulty in finding Nash equilibria, the fictitious play may provide another method to deal with multiagent reinforcement learning under SG framework. In fictitious play algorithm, the beliefs of other players' policies are represented by empirical distribution of their past play [1], [48]. Hence, the players only need to maintain their own Q values, which are related to joint actions and are weighted by their belief distribution of other players' actions.

Table V shows a fictitious play algorithm for two-player zero-sum SGs using a model [60].

For stationary policies of other players, the fictitious play algorithm becomes variants of individual Q-learning. For non-stationary policies of other players, these fictitious-play-based approaches have been empirically used in either competitive games where the players can model their adversarial opponents - called opponent modelling, or collaborative games where the players learn Q values of their joint actions - the player is called Joint Action Learner (JAL) [1].

For the fictitious-play-based approaches, the algorithms will converge to a Nash equilibrium in games that are iterated dominance solvable if all players are playing fictitious play [65].

Although the fictitious-play based learning eliminate the necessity of finding equilibria, learning agents have to model others and the learning convergence has to depend on some heuristic rules [61].

F. Multiagent SARSA Learning Algorithm

The Minimax-Q and Nash-Q learning algorithms are actually off-policy RL since they replace the *max* operator of individual Q-learning algorithm with their best response (Nash equilibrium policy). In RL, an off-policy learning algorithm always tries to converge to optimal Q values of optimal policy regardless of what policy is currently being executed. For off-policy learning algorithms, ε-greedy

TABLE V

THE FICTITIOUS PLAY ALGORITHM

1) Let $t \leftarrow 0$ and initialize Q_i for all $s \in \boldsymbol{\mathcal{S}}$ and $a_i \in \boldsymbol{\mathcal{A}}_i$.
2) Repeat for each state $s \in \boldsymbol{\mathcal{S}}$:
   a) Let $a_i = \arg\max_{a_i \in \boldsymbol{\mathcal{A}}_i} Q_i(s, a_i)$.
   b) Update $Q_i(s, a_i)$, $\forall s \in \boldsymbol{\mathcal{S}}, a_i^{'} \in \boldsymbol{\mathcal{A}}_i$

$$Q_i(s, a_i^{'}) \leftarrow Q_i(s, a_i^{'}) + R(s, < a_{-i}, a_i^{'} >) + \gamma \left(\Sigma_{s' \in \boldsymbol{\mathcal{S}}} T(s, a, s^{'}) V(s^{'}) \right)$$

   where,

$$V(s) = \max_{a_i \in \boldsymbol{\mathcal{A}}_i} Q_i(s, a_i)/t$$

3) $t \leftarrow t + 1$.

policies can usually be used to balance exploration and exploitation of learning space.

SARSA algorithm [31] is an on-policy RL algorithm that tries to converge to optimal Q values of policy currently being executed. Considering the disadvantages of the Minimax-Q and Nash-Q learning algorithms, a SARSA-based multi-agent algorithm called as EXORL (Extended Optimal Response Learning) was developed in [48]. In [48] the fact that the opponents may take stationary policies is taken into account, rather than Nash equilibrium policies. Once opponents take stationary policies, there is no need to finding Nash equilibria at all during learning. So, the learning updating can be simplified by eliminating the necessity of finding Nash equilibria if the opponents take stationary policies. In addition some heuristic rules were also employed to switch the algorithm between the Nash-equilibrium-based learning and the fictitious-play-based learning.

The EXORL algorithm is depicted as in Table VI. The basic idea of this algorithm is that the agent should learn a policy which is an optimal response to the opponent's policy, but it tries to reach a Nash equilibrium when the opponent is adaptable.

Like Nash-Q leaning algorithm, the EXORL algorithm will have a difficulty when there exist multiple equilibria. Another obvious shortcoming of the EXORL algorithm is that one agent is assumed to be capable of observing the opponent's action and rewards. In some cases this will be a very serious restriction since all the agents may learn their strategies simultaneously and one agent cannot obtain the actions of the opponent at all in advance. Moreover, the opponent also may take stochastic strategy instead of deterministic policies. The observing rewards obtained by the opponent will be more difficult since the rewards are only available after the policies are put into action practically.

Only some empirical results were given in [48] for the EXORL algorithm, there still lacks a theoretic foundation. Hence, a complete proof for the convergence will be expected. The theory results provided in [35], [39] may be helpful to obtain some convergence properties for the EXORL algorithm.

G. Policy Hill Climbing (PHC) Algorithm

The PHC algorithm updates Q values in the same way as the fictitious play algorithm, but it maintains a mixed policy (or stochastic policy) by performing hill-climbing in the space of mixed policies. Bowling and Veloso [60], [61] proposed a WoLF-PHC algorithm by adopting an idea of Win or Learn Fast

TABLE VI

THE EXORL ALGORITHM: A MULTIAGENT VERSION OF SARSA ALGORITHM

1) Initialize for all $\boldsymbol{s} \in \boldsymbol{\mathcal{S}}$, $\boldsymbol{a}_1 \in \boldsymbol{\mathcal{A}}_1$ and $\boldsymbol{a}_2 \in \boldsymbol{\mathcal{A}}_2$:

$$Q_1(\boldsymbol{s}, \boldsymbol{a}_1, \boldsymbol{a}_2) \leftarrow 0, \quad Q_2(\boldsymbol{s}, \boldsymbol{a}_1, \boldsymbol{a}_2) \leftarrow 0, \quad \pi_1(\boldsymbol{s}, \boldsymbol{a}_1) \leftarrow \frac{1}{|\boldsymbol{\mathcal{A}}_1|}, \quad \hat{\pi}_2(\boldsymbol{s}, \boldsymbol{a}_2) \leftarrow \frac{1}{|\boldsymbol{\mathcal{A}}_2|}$$

2) Repeat for each state $s \in \boldsymbol{\mathcal{S}}$:
   a) Choose action a_1^t according to $\pi_1(\boldsymbol{s}^t)$ with suitable exploration.
   b) Observe rewards (r_1^{t+1}, r_2^{t+1}), the action a_2^t taken by the opponent, and the next state s^{t+1}.
   c) Update value-functions for $i = 1, 2$

   $$Q_i(\boldsymbol{s}^{t-1}, \boldsymbol{a}_1^{t-1}, \boldsymbol{a}_2^{t-1}) \leftarrow (1-\alpha) Q_i(\boldsymbol{s}^{t-1}, \boldsymbol{a}_1^{t-1}, \boldsymbol{a}_2^{t-1}) + \alpha \left(r_i^t + \gamma Q_i(\boldsymbol{s}^t, \boldsymbol{a}_1^t, \boldsymbol{a}_2^t) \right)$$

   d) Update the estimate of opponent policy:

   $$\hat{\pi}_2(\boldsymbol{s}^t) \leftarrow (1 - \beta) + \beta \pi_2^t$$

   where π_2^t is a vector as follows

   $$\pi_2^t(a_2) = \begin{cases} 1 & \text{if } a_2 \neq a_2^t \\ 0 & \text{otherwise} \end{cases}$$

   e) Update policy $\pi_1(\boldsymbol{s}^{t-1})$ to maximize $O_{\boldsymbol{s}^{t-1}}(\pi_1(\boldsymbol{s}^{t-1}))$ defined as in the following:

   $$O_{\boldsymbol{s}}(\pi_1) = \pi_1^T Q_1(\boldsymbol{s}) \hat{\pi}_2(\boldsymbol{s}) - \sigma \rho_{\boldsymbol{s}}(\pi_1)$$

   where σ is a tuning parameter and,

   $$\rho_{\boldsymbol{s}}(\pi_1) = \max_{\pi_2} \left[\pi_1^T Q_2(\boldsymbol{s}) \pi_2(\boldsymbol{s}) - \pi_1^T Q_2(\boldsymbol{s}) \hat{\pi}_2(\boldsymbol{s}) \right]$$

(WoLF) and using a variable learning rate. The full algorithm for an agent i is shown in Table VII. The WoLF principle can result in the agent learning quickly when it is doing poorly and cautiously when it is performing well. The change in such a way for the learning rates will be helpful for convergence by not overfitting to the other agents' changing policies. At this point the WoLF-PHC algorithm seems to be attractive. Although many examples from MGs to zero-sum and general-sum SGs were given in [60], [61], a complete proof for the convergence properties has not been provided so far.

Rigorously speaking, the WoLF-PHC algorithm is still not a multiagent version of PHC algorithm since the learning factors of other agents in the non-Markovian environment are not taken into account at all. Thus, it is only rational and reasonable if the other agents are playing stationary strategies. In addition, the convergence may become very slow when the WoLF principle is applied [48].

TABLE VII

WoLF-PHC Algorithm for an agent i

1) Take learning rates $\alpha \in (0,1]$ and $\delta_l > \delta_w \in (0,1]$. Initialize $Q(\boldsymbol{s},\boldsymbol{a})$, $\pi(\boldsymbol{s},\boldsymbol{a})$, and $C(\boldsymbol{s})$ for all $s \in \boldsymbol{\mathcal{S}}$ and $a \in \boldsymbol{\mathcal{A}}_i$:
$$Q(\boldsymbol{s},\boldsymbol{a}) \leftarrow 0, \quad \pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow \frac{1}{|\boldsymbol{\mathcal{A}}_i|}, \quad C(\boldsymbol{s}) \leftarrow 0$$
2) Repeat for each state $s \in \boldsymbol{\mathcal{S}}$:
   a) Choose an action a at state s according to a mixed strategy $\pi(\boldsymbol{s},\boldsymbol{a})$ with suitable exploration.
   b) Observe reward r and new state $s^{'}$.
   c) Update value-function $Q(\boldsymbol{s},\boldsymbol{a})$, $\forall s \in \boldsymbol{\mathcal{S}}, a \in \boldsymbol{\mathcal{A}}_i$
$$Q(\boldsymbol{s},\boldsymbol{a}) \leftarrow (1-\alpha)Q(\boldsymbol{s},\boldsymbol{a}) + \alpha\left(r + \gamma V(s)\right)$$
   where,
$$V(s) = \max_{a^{'} \in \boldsymbol{\mathcal{A}}_i} Q(\boldsymbol{s}^{'},\boldsymbol{a}^{'})$$
   d) Update the estimate of average policy for all $a^{'} \in \boldsymbol{\mathcal{A}}_i$, $\bar{\pi}(\boldsymbol{s},\boldsymbol{a}^{'})$,
$$C(\boldsymbol{s}) \leftarrow C(\boldsymbol{s}) + 1$$
$$\bar{\pi}(\boldsymbol{s},\boldsymbol{a}^{'}) \leftarrow \bar{\pi}(\boldsymbol{s},\boldsymbol{a}^{'}) + (\pi(\boldsymbol{s},\boldsymbol{a}^{'}) - \bar{\pi}(\boldsymbol{s},\boldsymbol{a}^{'}))/C(\boldsymbol{s})$$
   e) Step $\pi(\boldsymbol{s},\boldsymbol{a})$ closer to the optimal policy: $\pi(\boldsymbol{s},\boldsymbol{a}) \leftarrow \pi(\boldsymbol{s},\boldsymbol{a}) + \Delta_{\boldsymbol{sa}}$ where,
$$\Delta_{\boldsymbol{sa}} = \begin{cases} -\delta_{\boldsymbol{sa}} & \text{if } a \neq \arg\max_a^{'} Q(\boldsymbol{s}^{'},\boldsymbol{a}^{'}) \\ \sum_{a^{'} \neq a} \delta_{\boldsymbol{sa}^{'}} & \text{otherwise} \end{cases}$$
$$\delta_{\boldsymbol{sa}} = \min\left(\pi(\boldsymbol{s},\boldsymbol{a}), \frac{\delta}{|\boldsymbol{\mathcal{A}}_i| - 1}\right)$$
$$\Delta = \begin{cases} \delta_w & \text{if } \Sigma_{a^{'} \in \boldsymbol{\mathcal{A}}_i} \pi(\boldsymbol{s},\boldsymbol{a}^{'}) Q(\boldsymbol{s},\boldsymbol{a}^{'}) > \Sigma_{a^{'} \in \boldsymbol{\mathcal{A}}_i} \pi(\bar{\boldsymbol{s}},\boldsymbol{a}^{'}) Q(\boldsymbol{s},\boldsymbol{a}^{'}) \\ \delta_l & \text{otherwise} \end{cases}$$
3) Until terminal state.

H. Other Algorithms

Sen et al. [66] studied multiagent coordination with learning classifier systems. Action policies mapping from perceptions to actions were used by multiple agents to learn coordination strategies without relying on shared information. The experimental results provided in [66] indicated that classifier systems can be more effective than the more widely used Q-learning scheme for multiagent coordination.

In multiagent systems, a learning agent may learn faster and establish some new rules for its own utility under future unseen situations if the experiences and knowledge from other agents are available to it. Considering this fact and the possible benefits gained from extracting proper rules out of the other agents' knowledge, a weighted strategy sharing (WSS) method was proposed in [46] for coordination learning by using the expertness of RL. In this method, each agent measures the expertness of the other agent in a team and assigns a weight to their knowledge and learns from them accordingly. Moreover, the Q-table of one of the cooperative agent is changed randomly.

In tackling the problem of coordination in multiagent systems, Boutilier [55] proposed a method for solving sequential multiagent decision problems by allowing agents to reason explicitly about specific coordination mechanisms. In this method an extension of value iteration in which the state space of the system is augmented with the state of the adopted coordination mechanism needs to be defined. This method allows the agents to reason about the short and long term prospects for coordination, and make decisions to engage or avoid coordination problems based on expected value [55].

A Bayesian approach to the coordination in multiagent RL was proposed in [47], [49]. Since this method requires a very restrictive assumption, i.e., the learning agent has the ability to observe the actions, states, and rewards of other agents, it is only rational to be used in coordination multiagent systems. The advantage of this method is that there is no need to find equilibria for obtaining a best response-based policy, like Minimax-Q or Nash-Q learning algorithms.

V. Scaling Reinforcement Learning to Multi-Robot Systems

Multi-robot learning is a challenge for learning to act in an non-Markovian environment which contains other robots. Robots in MRSs have to interact with and adapt to their environment, as well as learn from and adapt to their counterparts, rather than only taking stationary policies.

The tasks arising from MRSs have continuous state and/or action spaces. As a result, there will be difficulties in directly applying the aforementioned results on multiagent RL with finite states and actions to MRSs.

State and action abstraction approaches claim that extracting features from a large learning space are effective. The approaches include condition and behaviour extraction [2], teammate internal modelling, relationship-state estimation [5], and state vector quantisation [9]. However, all these approaches can be viewed as variants of individual Q-learning algorithms since they have modelled other robots either as parts of their environment or as stationary-policy holders.

One research on scaling reinforcement learning toward RoboCup soccer has been reported by Stone and Sutton [40]. The RoboCup soccer can be viewed as a special class of MRS and often be used a good test-bed for developing AI techniques in both single-agent and multiagent systems. The most challenging issues in MRSs also appear in the RoboCup soccer, such as the large state/action space, uncertainties, etc. In [40], an approach using episodic SMDP SARSA(λ) with linear tile-coding function approximation and variable λ was designed to learn higher-level decisions in a keepaway subtask of RobotCup soccer. Since the general theory of RL with function approximation has not yet been well understood, the linear SARSA(λ) which could be the best understood among current methods [40] was used in the scaling of reinforcement learning to RoboCup soccer. Moreover, they also claimed that it has advantages over off-policy methods such as Q-learning, which can be unstable with linear and other kinds of function approximation. However, they did not answer the open question that whether SARSA(λ) fails to converge as well.

To study the cooperation problems in learning many behaviours using RL, a subtask of RoboCup soccer, i.e., keep away was also investigated in [27] by combining SARSA(λ) and linear tile coding function approximation. However, only single-agent RL techniques, including SARSA(λ) with eligibility traces, tile coding function approximation, were directly applied to a multiagent domain. As pointed out previously, such a straightforward applications of single-agent RL techniques to multiagent systems have no sound theoretic foundation. Kostiadis and Hu [22] used Kanerva coding technique [31] to produce a decision-making module for possession football in RoboCup soccer. In this application Kanerva coding was used as a generalisation method to form a feature vector from raw sensory reading while the RL uses this feature vector to learn an optimal policy. Although the results provided in [22] demonstrated that the learning approach outperformed a number of benchmark policies including a hand-coded one, there lacked a theoretic analysis on how a series of single-agent RL techniques can work very well in a domain of multiagent systems.

The work in [2] presented a formulation of RL that enables learning in the concurrent multi-robot domain. The methodology adopted in that study makes use of behaviours and conditions to minimize the learning space. The credit assignment problem was dealt with through shaped reinforcement in the form of heterogeneous reinforcement functions and progress estimators.

Morales [64] proposed an approach to RL in robotics based on a relational representation. With this relational representation, this method can be applied over large search spaces and domain knowledge also can be incorporated. The main idea behind this approach is to represent states as sets of properties to characterize a particular state which may be common to other states. Since both states and actions are represented in terms of first order relations in the proposed framework of [64], policies are learned over such generalized representation.

In order to deal with the state space growing exponentially in the number of team members, Touzet [8] studied the robot awareness in cooperative mobile robot learning and proposed a method which requires a less cooperative mechanism, i.e., various levels of awareness rather than communication. The results illustrated in [8] with applications to the cooperative multi-robot observation of multiple moving targets shows some better performance than a purely collective learned behaviour.

In [11] a variety of methods were reviewed and used to demonstrate for learning in multi-robot domain. In that study behaviours were thought as the underlying control representation for handling scaling in learning policies and models, as well as learning from other agents. Touzet [16] proposed a pessimistic algorithm-based distributed lazy Q-learning for cooperative mobile robots. The pessimistic algorithm was used to compute a lower bound of the utility of executing an action in a given situation for each robot in a team. Although the Q-learning with lazy learning were used, the author also neglected the important fact for the applicability of Q-learning, that is in multi-agent systems the environment is not stationary.

Park et al. [23] studied modular Q-learning based multi-agent cooperation for robot soccer, where modular Q-learning was used to assign a proper action to an agent in multiagent systems. In this approach the architecture of modular Q-learning consists of learning modules and a mediator module. The function of the mediator is to select a proper action for the learning agent based on the Q-value obtained from each learning module.

Although there have been a variety of RL techniques that are developed for multiagent learning systems, very few of these techniques scale well to MRSs. On the one hand, the theory itself on multiagent RL systems in the finite discrete domains are still underway and have not been well-established. On the other hand, it is essentially very difficult to solve MRSs in general case because of the continuous and large state space as well as action space.

VI. Fuzzy Logic Systems and Multiagent Reinforcement Learning

Fuzzy Logic Controllers (FLCs) can be used to generalize Q-learning over continuous state spaces. The combination of FLCs with Q-learning has been proposed as Fuzzy Q-Learning (FQL) for many single robot applications [67]–[69].

In [70] a modular-fuzzy cooperative algorithm for multiagent systems was presented by taking advantage of modular architecture, internal model of other agent, and fuzzy logic in multiagent systems. In this algorithm, the internal model is used to estimate the agent's own action and evaluate other agents' actions. To overcome the problem of huge dimension of state space, fuzzy logic was used to map from input fuzzy sets representing the state space of each learning module to output fuzzy sets denoting action space. A fuzzy rule base of each learning module was built through the Q-learning, but without providing any convergence proof.

Kilic and Arslan [71] developed a Minimax fuzzy Q-learning for cooperative multi-agent systems. In this method, the learning agent always need to observe the actions other agents take and uses the Minimax Q-learning to update fuzzy Q-values by using fuzzy state and fuzzy goal representation. It should be noted that the Minimax Q-learning in [71] is from the sense of fuzzy operators (i.e., *max* and *min*) and it is totally different with the Minimax-Q learning of Littman [33]. Similarly to [70], there was no any proof to guarantee the optimal convergence of the Minimax fuzzy Q-learning.

A fuzzy game theoretic approach to multiagent coordination was presented in [72] by considering that utility values are usually approximate and the differences between utility values are somewhat vague. Thus, a fuzzy game theoretic approach may be useful when there are uncertains in utility values. For establishing a framework of fuzzy game, a series of notions, including fuzzy dominant relations, fuzzy Nash equilibrium, and fuzzy policies were defined in [72] under both fuzzy logic theory and game theory. It was also shown in [72] that a fuzzy strategy can outperform a mixed strategy in traditional game theory in tackling the cases of multiple equilibria, which is a very challenging issue in game theory. The study in [72] merely focused on one-stage policy fuzzy game, instead of RL. Thus, a combination of the fuzzy game theoretic approach with the popular RL techniques is quite possible for multi-agent/robot systems.

The convergence proof appears to be very difficult, particularly for the multiagent reinforcement learning with fuzzy generalizations. More recently, a convergence proof for single agent fuzzy reinforcement learning (FRL) was provided in [73]. However, one can find that the example presented in [73] does not reflect the theoretic work of that study at all. An obvious fact is that the triangular membership functions were used in the experiment instead of the Gaussian membership functions which are the basis of the theoretic work in [73]. Therefore one can find that the proving techniques and outcomes will be very difficult to extend to the domains of multiagent reinforcement learning with fuzzy logic generalizations. Furthermore, Watkins [29] also pointed out that Q-learning may not converge correctly for other representations rather than a look-up table representation for the Q-function.

VII. Main Challenges

MRSs often have all of the challenges for multiagent learning systems, such as continuous state, and action spaces, uncertainties, and nonstationary environment. Since the aforementioned algorithms in Section IV require the enumeration of states either for policies or value functions, one must have a major limitation for scaling the established multiagent reinforcement learning outcomes to MRs.

Most SGs studied in multiagent RL are of simple agent-based background where players execute perfect actions, observe complete states (or partially observed), and have full knowledge of other players' actions, states and rewards. This is not true for most MRSs. It is unfeasible for robots to completely obtain other players' information, especially for competitive games since opponents do not actively broadcast their information to share with the other players.

In addition, adversarial opponents may not act rationally. Accordingly, it is difficult to find Nash equilibria for the Nash-equilibrium-based approaches or model their dynamics for the fictitious-play-based approaches.

Taking into account the state of the art for multiagent learning system, there is particular difficulty in scaling the established (or partially recognized at least) multiagent RL algorithms, such as Minimax-Q leaning, Nash-Q learning, etc., to MRSs with large and continuous state and action spaces. On the one hand, most theoretic works on multiagent systems merely focus on the domains with small finite state and action sets. On the other hand there is still lacking of sound theoretic grounds which can be used to guide the scaling up the multiagent RL algorithms to MRSs. As a result, the learning performance (such as convergence, efficiency, and stability, etc.) cannot be guaranteed when approximation and generalization techniques are applied.

One important fact is that most of the multiagent RL algorithms, such as Minimax-Q leaning, Nash-Q learning is value-function based iteration method. Thus, for applying these technique to a continuous system the value-function has to be approximated by either using discretization or general approximators (such as neural networks,polynomial functions, fuzzy logic, etc.). However, some researchers has pointed out that the combination of DP methods with function approximators may produce unstable or divergent results even when applied to some very simple problems, see [74] as the references therein.

VIII. Future Research Directions

A. Coordination Games Or Teams Cooperation

Due to the aforementioned difficulties, a possible opportunity of RL in MRSs is to learn robot's coordination. Robots in a team may learn to work co-operatively or share their learned experience to accelerate their learning processes through their limited physical communication or observation abilities. These robots have common interests or identical payoffs. The games with these characteristics are referred to as co-ordination games. Littman [42] has proved the convergence of Q-learning in co-ordination games. However, it is hard to apply in practice because the conditions are difficult to verify in advance since games may contain adversarial equilibria and co-ordination equilibria simultaneously even though the FoF idea could be adopted. One special case is the team game where all players have the same Q values. Littman also showed that the conditions are easily satisfied in team games since there is only one MDP under these conditions. But the phenomenon of multiple equilibria in coordination games questions its practical values and remains a challenge issue. Thus an ongoing research is how to select one from multiple Nash equilibria. Furthermore, the exact same Q values are hard to maintain in each physical robot because of their sensory and executive uncertainties. Therefore, more research needs to be performed to gain a clear understanding of Q-learning convergence in the coordination games.

B. State and Action Abstraction of MRSs

Incomplete information, large learning space, and uncertainty are major obstacles for learning in MRSs. Learning in Behaviour-Based Robotics (BBR) can effectively reduce the search space in size and dimension and handle uncertainties locally. The action space will be transformed from continuous space of control inputs into some limited discrete sets. However, the convergence proof for the algorithms using state and action abstraction of MRSs will be a very challenging problem.

There are some advances in state and action abstraction of MRSs, though, there are still not completely satisfactory solutions to cope with continuous state and action spaces occurring in the domains of MRSs, for example, see at [64].

C. Generalization and Approximation

When the state and action spaces of the system are small and finite discrete, the lookup table method is generally feasible. However, in MRSs, the state and action spaces are often very huge or continuous, thus the lookup table method seems inappropriate. To solve this problem, besides the state and action abstraction, function approximation and generalization appears to be another feasible solution. For learning in a partially observable and nonstationary environment in the area of multiagent systems, Abul et al. [75] presented two multiagent based domain independent coordination mechanisms,i.e. perceptual coordination mechanism and observing coordination mechanism. The advantage of their approach is that multiple agents do not require explicit communication among themselves to learn coordinated behaviors. To cope with the huge state space, function approximation and generalization techniques were used in their work. Unfortunately, the proof of convergence with function approximation and generalization techniques was not provided at all in [75]. Currently, a generic theoretic framework for proving the optimal convergence of function approximation implementation of the popular RL algorithms (such as Q-learning) has not been established yet. Interestingly, there is an increasing effort in this direction in either single-agent or multi-agent systems. For the single-agent Temporal-Difference learning with linear function approximation, Tadić [76] studied the convergence and analyzed its asymptotic properties. Under mild conditions, the almost sure convergence of Temporal-Difference learning with linear function approximation was given and the upper error bound also can be determined.

D. Continuous Reinforcement Learning

Since there are many difficulties in extending RL of discrete finite domains to continuous learning systems, recently there have been a great deal of efforts towards directly developing continuous RL techniques for the complex continuous systems. Based on the theoretic framework of viscosity solutions, Munos [74] conducted a study of RL for the continuous state-space and time control problems. In this continuous case, the value-function will satisfy the Hamilton-Jacobi-Bellman (HJB) equation, which is a nonlinear first or second order equation. It is well known that solving the HJB equation is a very hard task. To solve the HJB equation, Munos used a powerful framework of viscosity solutions and showed that the unique value function solution to the HJB equation can be found in the sense of viscosity solutions.

A continuous Q-learning method was presented in [77] by using an incremental topology preserving map to partition the input space and the incorporation of bias to initialize the learning process. The resulting continuous action is an average of the discrete actions of the winning unit weighted by their Q-values [77]. More interesting, the author also showed the experimental results in robotics indicating that the continuous Q-learning method works better than the standard discrete action version of Q-learning in terms of both asymptotic performance and learning speed. This continuous Q-learning method still focus on single-agent systems. Hence a version of continuous Q-learning method for multiagent systems is expected accordingly.

Another ongoing research for solving continuous cases is continuous RL for feedback control systems [78], [79]. In [78] a continuous RL algorithm was developed and applied to the control problem involving the refinement of a Proportional-Integral (PI) controller. More interestingly, Tu [78] claimed that according to his results the continuous RL algorithm outperforms the discrete RL algorithms. In [79] continuous Q-Learning was studied for the use of RL methodology to the control of real systems. Linear Quadratic Regularization (LQR) techniques and Q-learning were combined for both linear and nonlinear continuous control systems.

IX. Other Related Work

A survey on the field of RL was given by Kaelbling et al. [80] from a computer-science perspective. Some central issues of RL, including trading off exploration and exploitation, learning from delayed reinforcement learning, making use of generalization, etc. were discussed in that investigation.

Stone and Veloso made a survey on multiahent systems from a machine learning perspective and presented a series of general multiagent scenarios. Although their investigation did not focus exclusively on robotic systems, robotic soccer was used as a test bed and several robotic multiagent systems were discussed. From the robot-soccer perspective, Kim and Vadakkepat [21] gave a survey on the multiagent systems and cooperative robotics. In their paper cooperative robotics and related issues on group architecture, resource conflict, and geometric problems were involved. However few theoretical research advances were mentioned.

A technical report on the SGs with multiple learning players in multiagent RL systems was given by Chalkiadakis [49]. However, there is almost no survey on the scaling up of multiagent RL to MRSs.

Shoham and Powers [58] made a very critical survey to the recent work in AI on multiagent RL. They argued that the work on learning in SGs is flawed, though there have been a lot of research in this field under the frameworks of SGs. The fundamental flaw, they thought, is unclarity about the problem or problems being addressed. In their work they questioned why focus on equilibria. They commented that the results concerning convergence of Nash-Q learning are quite awkward. Particularly, if multiple optimal equilibria exist then the agents need an oracle to coordinate their choices in order to converge to a Nash equilibrium, which begs the question of why to use learning for coordination at all [58]. Although there are many negative comments on the current work on RL in SGs, they also agreed that these results are unproblematic for the cases of zero-sum SGs and team or pure coordination games with common payoff functions.

X. Concluding Remarks

Recently there has been growing interests in scaling multiagent RL to MRSs. Although RL seems to be a good option for learning in multiagent systems, the continuous state and action spaces often hamper its applicability in MRSs. Fuzzy logic methodology seems to be a candidate for dealing with the approximation and generalization issues in the RL of multiagent systems. However, this scaling approach still remains open. Particularly there is a lack of theoretical grounds which can be used for proving the convergence and predicting performance of fuzzy logic-based multiagent RL (such as fuzzy multiagent Q-learning).

For cooperative robots systems, although some research outcomes in some special cases have been available now, there are also some difficulties (such as multiple equilibrium and selecting payoff structure, etc) for directly applying them to a practical MRS, e.g., robotic soccer system.

This paper gave a survey on multiagent RL in MRSs. The main objective of this work is to review some important advances in this field, though still not completely. Some challenging problems and promising research directions are provided and discussed. Although this paper cannot provide a complete and exhaustive survey of multiagent RL in MRSs, we still believe that it will help us more clearly understand the existing works and challenging issues in this ongoing research field.

ACKNOWLEDGMENT

This research is funded by the Engineering and Physical Sciences Research Council (EPSRC) under grant GR/S45508/01 (2003-2005).

REFERENCES

[1] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng, “Cooperative mobile robotics: antecedents and directions,” *Autonomous Robots*, vol. 4, pp. 1–23, 1997.

[2] M. J. Matarić, “Reinforcement learning in the multi-robot domain,” *Autonomous Robots*, vol. 4, pp. 73–83, 1997.

[3] F. Michaud and M. J. Matarić, “Learning from history for behavior-based mobile robots in non-stationary conditions,” *Autonomous Robots*, vol. 5, pp. 335–354, 1998.

[4] T. Balch and R. C. Arkin, “Behavior-based formation control for multirobot teams,” *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 926–939, December 1998.

[5] M. Asada, E. Uchibe, and K. Hosoda, “Co-operative behaviour acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development,” *Artificial Intelligence*, vol. 110, pp. 275–292, 1999.

[6] M. Wiering, R. P. Salustowicz, and J. Schmidhuber, “Reinforcement learning soccer teams with incomplete world models,” *Autonomous Robots*, vol. 7, pp. 77–88, 1999.

[7] S. V. Zwaan, J. A. A. Moreira, and P. U. Lima, “Cooperative learning and planning for multiple robots,” 2000. [Online]. Available: citeseer.nj.nec.com/299778.html

[8] C. F. Touzet, “Robot awareness in cooperative mobile robot learning,” *Autonomous Robots*, vol. 8, pp. 87–97, 2000.

[9] F. Fernandez and L. E. Parker, “Learning in large co-operative multi-robot domains,” *International Journal of Robotics and Automation*, vol. 16, no. 4, pp. 217–226, 2001.

[10] J. Liu and J. Wu, *Multi-Agent Robotic Systems*. CRC Press, 2001.

[11] M. J. Matarić, “Learning in behavior-based multi-robot systems: policies, models, and other agents,” *Journal of Cognitive Systems Research*, vol. 2, pp. 81–93, 2001.

[12] M. Bowling and M. Veloso, “Simultaneous adversarial multi-robot learning,” in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, August 2003.

[13] I. H. Elhajj, A. Goradia, N. Xi, and et al, “Design and analysis of internet-based tele-coordinated multi-robot systems,” *Autonomous Robots*, vol. 15, pp. 237–254, 2003.

[14] L. Iocchi, D. Nardi, M. Piaggio, and A. Sgorbissa, “Distributed coordination in heterogeneous multi-robot systems,” *Autonomous Robots*, vol. 15, pp. 155–168, 2003.

[15] M. J. Mataric, G. S. Sukhatme, and E. H. Østergaard, “Multi-robot task allocation in uncertain environments,” *Autonomous Robots*, vol. 14, pp. 255–263, 2003.

[16] C. F. Touzet, “Distributed lazy Q-learning for cooperative mobile robots,” *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, pp. 5–13, 2004.

[17] T. Fujii, Y. Arai, H. Asama, and I. Endo, “Multilayered reinforcement learning for complicated collision avoidance problems,” in *Proceeding of the IEEE International Conference on Robotics and Automation*, 1998, pp. 2186–2191.

[18] R. P. Salustowicz, M. A. Wiering, and J. Schmidhuber, “Learning team strategies: soccer case studies,” *Machine Learning*, vol. 33, pp. 263–282, 1998.

[19] S. Sen, “Multiagent systems: Milestones and new horizons.” [Online]. Available: citeseer.ist.psu.edu/294585.html

[20] G. Weiß and P. Dillenbourg, *What is ’multi’ in multi-agent learning?*, 1999, ch. 4 in Collaborative-learning: Cognitive. [Online]. Available: wwwbrauer.in.tum.de/˜weissg/Docs/weiss-dillenbourg.pdf

[21] J.-H. Kim and P. Vadakkepat, “Multi-agent systems: a survey from the robot-soccer perspective,” *International Journal of Intelligent Automation and Soft Computing*, vol. 6, no. 1, pp. 3–17, 2000.

[22] K. Kostiadis and H. Hu, “KaBaGe-RL: kanerva-based generalisation and reinforcement learning for possession football,” in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hawaii, 2001.

[23] K.-H. Park, Y.-J. Kim, and J.-H. Kim, “Modular Q-learning based multi-agent cooperation for robot soccer,” *Robotics and Autonomous Systems*, vol. 35, pp. 109–122, 2001.

[24] A. Merke and M. Riedmiller, “A reinforcement learning approach to robotic soccer,” in *RoboCup 2001*, 2001, pp. 435–440.

[25] S. Maes, K. Tuyls, and B. Manderick, “Reinforcement learning in large state spaces: simulated robotic soccer as a testbed,”

2002. [Online]. Available: http://citeseer.ist.psu.edu/502799.html

[26] W. D. Smart and L. P. Kaelbling, "Effective reinforcement learning for mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2002. [Online]. Available: www.ai.mit.edu/people/lpk/papers/icra2002.pdf

[27] S. Valluri and S. Babu, "Reinforcement learning for keepaway soccer problem," 2002. [Online]. Available: http://www.cis.ksu.edu/~babu/final/html/ProjectReport.htm

[28] M. Wahab, "Reinforcement learning in multiagent systems," 2003. [Online]. Available: http://www.cs.mcgill.ca/~mwahab/RL%20in%20MAS.pdf

[29] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[30] M. E. Harmon and S. S. Harmon, "Reinforcement learning: a tutorial," 1996. [Online]. Available: http://citeseer.ist.psu.edu/harmon96reinforcement.html

[31] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, Massachusetts: MIT Press, 1998.

[32] M. Tan, "Multi-agent reinforcement learning: independent vs. cooperative agents," in *Proceedings of the Tenth International Conference on Machine Learning*, Amherst, MA, 1993, pp. 330–337.

[33] M. L. Littman, "Markov games as a framework for multi-agent learning," in *Proceedings of the Eleventh International Conference on Machine Learning*, San Francisco, California, 1994, pp. 157–163.

[34] M. L. Littman and C. Szepesvári, "A generalized reinforcement-learning model: convergence and applications," in *Proceedings of the Thirteenth International Conference on Machine Learning*, Bari, Italy, July 3-6 1996, pp. 310–318.

[35] C. Szepesvári and M. L. Littman, "Generalized markov decision processes: dynamic-programming and reinforcement-learning algorithms," Department of Computer Science, Brown University, Technical report CS-96-11, November 1996. [Online]. Available: http://www.cs.duke.edu/~mlittman/docs/unrefer.html

[36] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in coopertive multiagent systems," in *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, Madison, WI, 1998, pp. 746–752.

[37] J. Hu and M. P. Wellman, "Multiagent reinforcement learning in stochastic games," 1999. [Online]. Available: citeseer.ist.psu.edu/hu99multiagent.html

[38] R. Sun and D. Qi, "Rationality assumptions and optimality of co-learning," in *Design and Applications of Intelligent Agents, Third Pacific Rim International Workshop on Multi-Agents, PRIMA 2000*, ser. Lecture Notes in Computer Science, C. Zhang and V.-W. Soo, Eds., vol. 1881. Springer, 2000, pp. 61–75.

[39] C. Szepesvári and M. L. Littman, "A unified analysis of value-function-based reinforcement learning algorithms," *Neural Computing*, vol. 11, no. 8, pp. 2017–2059, 1999.

[40] P. Stone and M. Veloso, "Multiagent systems: a survey from a machine learning perspective," *Autonomous Robots*, vol. 8, pp. 345–383, 2000.

[41] J. Hu and M. P. Wellman, "Learning about other agents in a dynamic multiagent system," *Journal of Cognitive Systems Research*, vol. 2, pp. 67–79, 2001.

[42] M. L. Littman, "Value-function reinforcement learning in markov games," *Journal of Cognitive Systems Research*, vol. 2, pp. 55–66, 2001.

[43] M. L. Littman and P. Stone, "Leading best-response strategies in repeated games," in *The 17th Annual International Joint Conference on Artificial Intelligence Workshop on Economic Agents, Models, and Mechanism*, 2001.

[44] M. L. Littman, "Friend-or-foe Q-learning in general-sum games," in *Proceedings of The 18th International Conference on Machine Learning*, Morgan Kaufman, 2001, pp. 322–328.

[45] F. A. Dahl, "The lagging anchor algorithm: reinforcement learning in two-player zero-sum games with imperfect information," *Machine Learning*, vol. 49, pp. 5–37, 2002.

[46] M. N. Ahmadabali and M. Asadpour, "Expertness based cooperative Q-learning," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 32, no. 1, pp. 66–76, February 2002.

[47] G. Chalkiadakis, "Multiagent reinforcement learning: stochastic games with multiple learning players," Department of Computer Science, Univeristy of Toronto," Technical report, March 2003. [Online]. Available: www.cs.toronto.edu/~gehalk/DepthReport/DepthReport.ps

[48] N. Suematsu and A. Hayashi, "A multiagent reinforcement learning algorithm using extended optimal response," in *Proceedings of the First International Joint Conference on Autonomous Agents & Multiagent Systems*, Bologna, Italy, July 15-19 2002, pp. 370–377.

[49] G. Chalkiadakis and C. Boutilier, "Multiagent reinforcement learning: theoretical framework and an algorithm," in *the Second International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS)*, Melbourne, Australia, July 14-18 2003, pp. 709–716.

[50] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *Journal of Machine Learning Research*, vol. 4, pp. 1039–1069, 2003.

[51] S. Singh, T. Jaakkola, M. L. Littman, and et al, "Convergence results for single-step on-policy reinforcement-learning algorithms," *Machine Learning*, vol. 39, pp. 287–308, 2000.

[52] T. Basar and G. J. Olsder, *Dynamic Noncooperative Game Theory*. London: Academic Press, 1982.

[53] B. Banerjee and J. Peng, "Adaptive policy gradient in multiagent learning," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. ACM Press, 2003, pp. 686–692.

[54] D. Fudenberg and J. Tirole, *Game Theory*. London: MIT Press, 1991.

[55] C. Boutilier, "Sequential optimality and coordination in multi-

agent systems," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999, pp. 478–485.

[56] M. G. Lagoudakis and R. Parr, "Value function approximation in zero-sum markov games," 2002. [Online]. Available: http://www.cs.duke.edu/˜mgl/papers/PDF/uai2002.pdf

[57] X. Li, "Refining basis functions in least-square approximation of zero-sum markov games," 2003. [Online]. Available: http://www.xiaolei.org/research/li03basis.pdf

[58] Y. Shoham and R. Powers, "Multi-agent reinforcement learning: a critical survey," 2003. [Online]. Available: http://www.stanford.edu/˜grenager/MALearning˙ACriticalSurvey˙2003˙0516.%pdf

[59] J. Hu and M. P. Wellman, "Multiagent reinforcement learning: theoretical framework and an algorithm," in *Proceedings of the Fifteenth International Conference on Machine Learning*, San Francisco, California, 1998, pp. 242–250.

[60] M. Bowling, "Multiagent learning in the presence of agents with limitations," Ph.D. dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, May 2003, CMU-CS-03-118.

[61] M. H. Bowling and M. M. Veloso, "Multiagent learning using a variable learning rate," *Artificial Intelligence*, vol. 136, no. 2, pp. 215–250, 2002. [Online]. Available: citeseer.ist.psu.edu/bowling02multiagent.html

[62] B. Banerjee and J. Peng, "Convergent gradient ascent in general-sum games," in *Proceedings of the 13th European Conference on Machine Learning*, August 13-19 2002, pp. 686–692.

[63] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Proceeesing Systems*. MIT Press, 12, pp. 1057–1063.

[64] E. F. Morales, "Scaling up reinforcement learning with a relational representation," in *Workshop on Adaptability in Multi-Agent Systems,The First RoboCup Australian Open 2003 (AORC-2003)*, Sydney, Australia, January 31 2003.

[65] D. Fudenberg and D. K. Levine, *The Theory of Learning in Games*. Cambridge, Massachusetts: MIT Press, 1999.

[66] S. Sen and M. Sekaran, "Multiagent coordination with learning classifier systems," in *Proceedings of the IJCAI Workshop on Adaption and Learning in Multi-Agent Systems*, G. Weiß and S. Sen, Eds., vol. 1042. Springer Verlag, 1996, pp. 218–233. [Online]. Available: citeseer.ist.psu.edu/sen96multiagent.html

[67] H. R. Berenji and D. A. Vengerov, "Cooperation and coordination between fuzzy reinforcement learning agents in continuous-state partially observable markov decision processes," in *Proceedings of the 8th IEEE International Conference onFuzzy Systems (FUZZ-IEEE'99)*, 2000.

[68] ——, "Advantages of cooperation between reinforcement learning agents in difficult stochastic problems," in *Proceedings of the 9th IEEE International Conference onFuzzy Systems (FUZZ-IEEE '00)*, 2000.

[69] H. R. Berenji and D. Vengerov, "Generalized markov decision processes: dynamic-programming and reinforcement-learning algorithms," Intelligent Inference Systems Corp.,Sunnyvale, CA, Technical report IIS-00-10, October 2000. [Online]. Available: http://www.iiscorp.com/projects/multi-agent/tech˙rep˙iis˙00˙10.pdf

[70] I. Gültekin and A. Arslan, "Modular-fuzzy cooperative algorithm for multi-agent systems," in *Advances in Information Systems, Second International Conference, ADVIS 2002, Izmir, Turkey, October 23-25, 2002, Proceedings*, ser. Lecture Notes in Computer Science, T. M. Yakhno, Ed., vol. 2457. Springer, 2002, pp. 255–263.

[71] A. Kilic and A. Arslan, "Minimax fuzzy Q-learning in cooperative multi-agent systems," in *Advances in Information Systems, Second International Conference, ADVIS 2002, Izmir, Turkey, October 23-25, 2002, Proceedings*, ser. Lecture Notes in Computer Science, T. M. Yakhno, Ed., vol. 2457. Springer, 2002, pp. 264–272.

[72] S.-H. Wu and V.-W. Soo, "Fuzzy game theoretic approach to multi-agent coordination," in *Multiagent Platforms, First Pacific Rim International Workshop on Multi-Agents, PRIMA '98, Singapore, November 23, 1998, Selected Papers*, ser. Lecture Notes in Computer Science, T. Ishida, Ed., vol. 1599. Springer, 1999, pp. 76–87.

[73] H. R. Berenji and D. Vengerov, "A convergent ActorCCritic-based FRL algorithm with application to power management of wireless transmitters," *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 4, pp. 478–485, August 2003.

[74] R. Munos, "A study of reinforcement learning in the continuous case by the means of viscosity solutions," *Machine Learning*, vol. 40, pp. 265–299, 2000.

[75] O. Abul, F. Polat, and R. Alhajj, "Multiagent reinforcement learning using function approximation," *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Application and Reviews*, vol. 30, no. 4, pp. 485–497, November 2000.

[76] V. Tadić, "On the convergence of temporal-difference learning with linear function approximation," *Machine Learning*, vol. 42, pp. 241–267, 2001.

[77] J. Del R. Millán, D. Posenato, and E. Dedieu, "Continuous-action Q-learning," *Machine Learning*, vol. 49, no. 2-3, 2002.

[78] J. Tu, "Continuous reinforcement learning for feedback control systems," Master's thesis, Computer Science Department, Colorado State University, Fort Collins, Colorado, 2001. [Online]. Available: http://www.engr.colostate.edu/nnhvac/papers/jilin-tu.ps.gz

[79] S. H. G. ten Hagen, "Continuous state space Q-Learning for control of nonlinear systems," Ph.D. dissertation, Faculty of Science, IAS, University of Amsterdam, Kruislaan 403, 1098 SJ Amsterdam, 2001.

[80] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: a survey," *Journal of Artificial Intelligence Research*, vol. 4, no. 4, pp. 237–285, 1996.