A Comprehensive Survey of Multiagent Reinforcement Learning

Lucian Buşoniu, Robert Babuška, and Bart De Schutter

***Abstract*—Multiagent systems are rapidly finding applications in a variety of domains, including robotics, distributed control, telecommunications, and economics. The complexity of many tasks arising in these domains makes them difficult to solve with preprogrammed agent behaviors. The agents must, instead, discover a solution on their own, using learning. A significant part of the research on multiagent learning concerns reinforcement learning techniques. This paper provides a comprehensive survey of multiagent reinforcement learning (MARL). A central issue in the field is the formal statement of the multiagent learning goal. Different viewpoints on this issue have led to the proposal of many different goals, among which two focal points can be distinguished: stability of the agents' learning dynamics, and adaptation to the changing behavior of the other agents. The MARL algorithms described in the literature aim—either explicitly or implicitly—at one of these two goals or at a combination of both, in a fully cooperative, fully competitive, or more general setting. A representative selection of these algorithms is discussed in detail in this paper, together with the specific issues that arise in each category. Additionally, the benefits and challenges of MARL are described along with some of the problem domains where the MARL techniques have been applied. Finally, an outlook for the field is provided.**

***Index Terms*—Distributed control, game theory, multiagent systems, reinforcement learning.**

I. Introduction

A MULTIAGENT system [1] can be defined as a group of autonomous, interacting entities sharing a common environment, which they perceive with sensors and upon which they act with actuators [2]. Multiagent systems are finding applications in a wide variety of domains including robotic teams, distributed control, resource management, collaborative decision support systems, data mining, etc. [3], [4]. They may arise as the most natural way of looking at the system, or may provide an alternative perspective on systems that are originally regarded as centralized. For instance, in robotic teams, the control authority is naturally distributed among the robots [4]. In resource management, while resources can be managed by a central authority, identifying each resource with an agent may provide a helpful, distributed perspective on the system [5].

Manuscript received November 10, 2006; revised March 7, 2007 and June 18, 2007. This work was supported by the Senter, Ministry of Economic Affairs of The Netherlands, under Grant BSIK03024 within the BSIK project "Interactive Collaborative Information Systems." This paper was recommended by Associate Editor J. Lazansky.

L. Busoniu and R. Babusaka are with the Delft Center for Systems and Control, Faculty of Mechanical Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: i.l.busoniu@tudelft.nl; r.babuska@tudelft.nl).

B. De Schutter is with the Delft Center for Systems and Control, Faculty of Mechanical Engineering and also with the Marine and Transport Technology Department, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: b@deschutter.info).

Digital Object Identifier 10.1109/TSMCC.2007.913919

Although the agents in a multiagent system can be programmed with behaviors designed in advance, it is often necessary that they learn new behaviors online, such that the performance of the agent or of the whole multiagent system gradually improves [4], [6]. This is usually because the complexity of the environment makes the *a priori* design of a good agent behavior difficult, or even, impossible. Moreover, in an environment that changes over time, a hardwired behavior may become inappropriate.

A reinforcement learning (RL) agent learns by trial-and-error interaction with its dynamic environment [6]–[8]. At each time step, the agent perceives the complete state of the environment and takes an action, which causes the environment to transit into a new state. The agent receives a scalar reward signal that evaluates the quality of this transition. This feedback is less informative than in supervised learning, where the agent would be given the correct actions to take [9] (such information is, unfortunately, not always available). The RL feedback is, however, more informative than in unsupervised learning, where the agent would be left to discover the correct actions on its own, without any explicit feedback on its performance [10]. Well-understood algorithms with good convergence and consistency properties are available for solving the single-agent RL task, both when the agent knows the dynamics of the environment and the reward function (the task model), and when it does not. Together with the simplicity and generality of the setting, this makes RL attractive also for multiagent learning. However, several new challenges arise for RL in multiagent systems. Foremost among these is the difficulty of defining a good learning goal for the multiple RL agents. Furthermore, most of the times each learning agent must keep track of the other learning (and therefore, nonstationary) agents. Only then will it be able to coordinate its behavior with theirs, such that a coherent joint behavior results. The nonstationarity also invalidates the convergence properties of most single-agent RL algorithms. In addition, the scalability of algorithms to realistic problem sizes, already problematic in single-agent RL, is an even greater cause for concern in multiagent reinforcement learning (MARL).

The MARL field is rapidly expanding, and a wide variety of approaches to exploit its benefits and address its challenges have been proposed over the last few years. These approaches integrate developments in the areas of single-agent RL, game theory, and more general, direct policy search techniques. The goal of this paper is to provide a comprehensive review of MARL. We thereby select a representative set of approaches that allows us to identify the structure of the field, to provide insight into the current state of the art, and to determine some important directions for future research.

1094-6977/$25.00 © 2008 IEEE

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.

A. Contribution and Related Work

This paper provides a detailed discussion of the MARL techniques for fully cooperative, fully competitive, and mixed (neither cooperative nor competitive) tasks. The focus is placed on autonomous multiple agents learning how to solve dynamic tasks online, using learning techniques with roots in dynamic programming and temporal-difference RL. Different viewpoints on the central issue of the learning goal in MARL are discussed. A classification of the MARL algorithms along several taxonomy dimensions is also included. In addition, we provide an overview of the challenges and benefits in MARL, and of the problem domains where the MARL techniques have been applied. We identify a set of important open issues and suggest promising directions to address these issues.

Besides single-agent RL, MARL has strong connections with game theory, evolutionary computation, and optimization theory, as will be outlined next.

Game theory—the study of multiple interacting agents trying to maximize their rewards [11]—and, especially, the theory of learning in games [12], make an essential contribution to MARL. We focus here on algorithms for dynamic multiagent tasks, whereas most game-theoretic results deal with static (stateless) one-shot or repeated tasks. We investigate the contribution of game theory to the MARL algorithms for dynamic tasks, and review relevant game-theoretic algorithms for static games. Other authors have investigated more closely the relationship between game theory and MARL. Bowling and Veloso [13] discuss several MARL algorithms, showing that these algorithms combine temporal difference RL with game-theoretic solvers for the static games arising in each state of the dynamic environment. Shoham *et al.* [14] provide a critical evaluation of the MARL research, and review a small set of approaches that are representative for their purpose.

Evolutionary computation applies principles of biological evolution to the search for solutions of the given task [15], [16]. Populations of candidate solutions (agent behaviors) are stored. Candidates are evaluated using a fitness function related to the reward, and selected for breeding or mutation on the basis of their fitness. Since we are interested in online techniques that exploit the special structure of the RL task by learning a value function, we do not review here evolutionary learning techniques. Evolutionary learning, and in general, direct optimization of the agent behaviors, cannot readily benefit from the RL task structure. Panait and Luke [17] offer a comprehensive survey of evolutionary learning, as well as MARL, but only for cooperative agent teams. For the interested reader, examples of coevolution techniques, where the behaviors of the agents evolve in parallel, can be found in [18]–[20]. Complementary, team learning techniques, where the entire set of agent behaviors is discovered by a single evolution process, can be found, e.g., in [21]–[23].

Evolutionary multiagent learning is a special case of a larger class of techniques originating in optimization theory that explore directly the space of agent behaviors. Other examples in this class include gradient search [24], probabilistic hill climbing [25], and even more general behavior modification heuristics [26]. The contribution of direct policy search to the MARL algorithms is discussed in this paper, but general policy search techniques are not reviewed. This is because, as stated before, we focus on techniques that exploit the structure of the RL problem by learning value functions.

Evolutionary game theory sits at the intersection of evolutionary learning and game theory [27]. We discuss only the contribution of evolutionary game theory to the analysis of multiagent RL dynamics. Tuyls and Nowé [28] investigate the relationship between MARL and evolutionary game theory in more detail, focusing on static tasks.

B. Overview

The remainder of this paper is organized as follows. Section II introduces the necessary background in single-agent and multiagent RL. Section III reviews the main benefits of MARL and the most important challenges that arise in the field, among which is the definition of an appropriate formal goal for the learning multiagent system. Section IV discusses the formal goals put forward in the literature, which consider stability of the agent's learning process and adaptation to the dynamic behavior of the other agents. Section V provides a taxonomy of the MARL techniques. Section VI reviews a representative selection of the MARL algorithms, grouping them by the type of targeted learning goal (stability, adaptation, or a combination of both) and by the type of task (fully cooperative, fully competitive, or mixed). Section VII then gives a brief overview of the problem domains where MARL has been applied. Section VIII distills an outlook for the MARL field, consisting of important open questions and some suggestions for future research. Section IX concludes and closes the paper.

Note that algorithm names are typeset in italics throughout the paper, e.g., *Q-learning*.

II. Background: Reinforcement Learning

In this section, the necessary background on single-agent and multiagent RL is introduced [7], [13]. First, the single-agent task is defined and its solution is characterized. Then, the multiagent task is defined. Static multiagent tasks are introduced separately, together with necessary game-theoretic concepts. The discussion is restricted to finite state and action spaces, as the large majority of MARL results is given for finite spaces.

A. Single-Agent Case

In single-agent RL, the environment of the agent is described by a Markov decision process.

Definition 1: A finite *Markov decision process* is a tuple $\langle X, U, f, \rho \rangle$ where X is the finite set of environment states, U is the finite set of agent actions, $f : X \times U \times X \to [0, 1]$ is the state transition probability function, and $\rho : X \times U \times X \to \mathbb{R}$ is the reward function.[1]

The state signal $x_k \in X$ describes the environment at each discrete time-step k. The agent can alter the state at each time

[1]Throughout the paper, the standard control-theoretic notation is used: x for state, X for state space, u for control action, U for action space, f for environment (process) dynamics. We denote reward functions by ρ, to distinguish them from the instantaneous rewards r and the returns R. We denote agent policies by h.

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.

step by taking actions $u_k \in U$. As a result of the action u_k, the environment changes its state from x_k to some $x_{k+1} \in X$ according to the state transition probabilities given by f: the probability of ending up in x_{k+1} given that u_k is executed in x_k is $f(x_k, u_k, x_{k+1})$. The agent receives a scalar reward $r_{k+1} \in \mathbb{R}$, according to the reward function ρ: $r_{k+1} = \rho(x_k, u_k, x_{k+1})$. This reward evaluates the immediate effect of action u_k, i.e., the transition from x_k to x_{k+1}. It says, however, nothing directly about the long-term effects of this action.

For deterministic models, the transition probability function f is replaced by a simpler transition function, $\bar{f} : X \times U \to X$. It follows that the reward is completely determined by the current state and action: $r_{k+1} = \bar{\rho}(x_k, u_k)$, $\bar{\rho} : X \times U \to \mathbb{R}$.

The behavior of the agent is described by its policy h, which specifies how the agent chooses its actions given the state. The policy may be either stochastic, $h : X \times U \to [0, 1]$, or deterministic, $\bar{h} : X \to U$. A policy is called stationary if it does not change over time.

The agent's goal is to maximize, at each time-step k, the expected discounted return

$$R_k = E\left\{\sum_{j=0}^{\infty} \gamma^j r_{k+j+1}\right\} \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor, and the expectation is taken over the probabilistic state transitions. The quantity R_k compactly represents the reward accumulated by the agent in the long run. Other possibilities of defining the return exist [8]. The discount factor γ can be regarded as encoding increasing uncertainty about rewards that will be received in the future, or as a means to bound the sum that otherwise might grow infinitely.

The task of the agent is, therefore, to maximize its long-term performance, while only receiving feedback about its immediate, one-step performance. One way it can achieve this is by computing an optimal action-value function.

The *action-value function* (Q-function), $Q^h : X \times U \to \mathbb{R}$, is the expected return of a state-action pair given the policy h: $Q^h(x, u) = E\{\sum_{j=0}^{\infty} \gamma^j r_{k+j+1} \mid x_k = x, u_k = u, h\}$. The optimal Q-function is defined as $Q^*(x, u) = \max_h Q^h(x, u)$. It satisfies the Bellman optimality equation

$$Q^*(x, u) = \sum_{x' \in X} f(x, u, x')\left[\rho(x, u, x') + \gamma \max_{u'} Q^*(x', u')\right] \qquad \forall x \in X, \quad u \in U. \tag{2}$$

This equation states that the optimal value of taking u in x is the expected immediate reward plus the expected (discounted) optimal value attainable from the next state (the expectation is explicitly written as a sum since X is finite).

The greedy policy is deterministic and picks for every state the action with the highest Q-value

$$\bar{h}(x) = \arg\max_u Q(x, u). \tag{3}$$

The agent can achieve the learning goal by first computing Q^* and then choosing actions by the greedy policy, which is optimal (i.e., maximizes the expected return) when applied to Q^*.

A broad spectrum of single-agent RL algorithms exists, e.g., model-based methods based on dynamic programming [29]–[31], model-free methods based on online estimation of value functions [32]–[35], and model-learning methods that estimate a model, and then learn using model-based techniques [36], [37]. Most MARL algorithms are derived from a model-free algorithm called *Q-learning* [32], e.g., [13], [38]–[42].

Q-learning [32] turns (2) into an iterative approximation procedure. The current estimate of Q^* is updated using estimated samples of the right-hand side of (2). These samples are computed using actual experience with the task, in the form of rewards r_{k+1} and pairs of subsequent states x_k, x_{k+1}

$$Q_{k+1}(x_k, u_k) = Q_k(x_k, u_k) + \alpha_k\left[r_{k+1} + \gamma \max_{u'} Q_k(x_{k+1}, u') - Q_k(x_k, u_k)\right]. \tag{4}$$

Since (4) does not require knowledge about the transition and reward functions, *Q-learning* is model-free. The learning rate $\alpha_k \in (0, 1]$ specifies how far the current estimate $Q_k(x_k, u_k)$ is adjusted toward the update target (sample) $r_{k+1} + \gamma \max_{u'} Q(x_{k+1}, u')$. The learning rate is typically time varying, decreasing with time. Separate learning rates may be used for each state-action pair. The expression inside the square brackets is the temporal difference, i.e., the difference between the estimates of $Q^*(x_k, u_k)$ at two successive time steps, $k + 1$ and k.

The sequence Q_k provably converges to Q^* under the following conditions [32], [43], [44].

1) Explicit, distinct values of the Q-function are stored and updated for each state-action pair.
2) The time series of learning rates used for each state-action pair sums to infinity, whereas the sum of its squares is finite.
3) The agent keeps trying all actions in all states with nonzero probability.

The third condition means that the agent must sometimes explore, i.e., perform other actions than dictated by the current greedy policy. It can do that, e.g., by choosing at each step a random action with probability $\varepsilon \in (0, 1)$, and the greedy action with probability $(1 - \varepsilon)$. This is ε-greedy exploration. Another option is to use the Boltzmann exploration strategy, which in state x selects action u with probability

$$h(x, u) = \frac{e^{Q(x,u)/\tau}}{\sum_{\tilde{u}} e^{Q(x,\tilde{u})/\tau}} \tag{5}$$

where $\tau > 0$, the temperature, controls the randomness of the exploration. When $\tau \to 0$, this is equivalent with greedy action selection (3). When $\tau \to \infty$, action selection is purely random. For $\tau \in (0, \infty)$, higher-valued actions have a greater chance of being selected than lower-valued ones.

B. Multiagent Case

The generalization of the Markov decision process to the multiagent case is the stochastic game.

Definition 2: A *stochastic game* (SG) is a tuple $\langle X, U_1, \ldots, U_n, f, \rho_1, \ldots, \rho_n \rangle$ where n is the number of agents, X is the

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.

discrete set of environment states, U_i, $i = 1, \ldots, n$ are the discrete sets of actions available to the agents, yielding the joint action set $\boldsymbol{U} = U_1 \times \cdots \times U_n$, f: $X \times \boldsymbol{U} \times X \to [0, 1]$ is the state transition probability function, and ρ_i: $X \times \boldsymbol{U} \times X \to \mathbb{R}$, $i = 1, \ldots, n$ are the reward functions of the agents.

In the multiagent case, the state transitions are the result of the joint action of all the agents, $\boldsymbol{u}_k = [u_{1,k}^{\mathrm{T}}, \ldots, u_{n,k}^{\mathrm{T}}]^{\mathrm{T}}, \boldsymbol{u}_k \in \boldsymbol{U}$, $u_{i,k} \in U_i$ (T denotes vector transpose). Consequently, the rewards $r_{i,k+1}$ and the returns $R_{i,k}$ also depend on the joint action. The policies h_i: $X \times U_i \to [0, 1]$ form together the joint policy $\boldsymbol{h}$. The Q-function of each agent depends on the joint action and is conditioned on the joint policy, $Q_i^{\boldsymbol{h}}$: $X \times \boldsymbol{U} \to \mathbb{R}$.

If $\rho_1 = \cdots = \rho_n$, all the agents have the same goal (to maximize the same expected return), and the SG is fully cooperative. If $n = 2$ and $\rho_1 = -\rho_2$, the two agents have opposite goals, and the SG is fully competitive.[2] Mixed games are stochastic games that are neither fully cooperative nor fully competitive.

C. Static, Repeated, and Stage Games

Many MARL algorithms are designed for static (stateless) games, or work in a stagewise fashion, looking at the static games that arise in each state of the stochastic game. Some game-theoretic definitions and concepts regarding static games are, therefore, necessary to understand these algorithms [11], [12].

A *static (stateless) game* is a stochastic game with $X = \emptyset$. Since there is no state signal, the rewards depend only on the joint actions ρ_i: $\boldsymbol{U} \to \mathbb{R}$. When there are only two agents, the game is often called a bimatrix game, because the reward function of each of the two agents can be represented as a $|U_1| \times |U_2|$ matrix with the rows corresponding to the actions of agent 1, and the columns to the actions of agent 2, where $|\cdot|$ denotes set cardinality. Fully competitive static games are also called zero-sum games, because the sum of the agents' reward matrices is a zero matrix. Mixed static games are also called general-sum games, because there is no constraint on the sum of the agents' rewards.

When played repeatedly by the same agents, the static game is called a repeated game. The main difference from a one-shot game is that the agents can use some of the game iterations to gather information about the other agents or the reward functions, and make more informed decisions thereafter. A *stage game* is the static game that arises when the state of an SG is fixed to some value. The reward functions of the stage game are the expected returns of the SG when starting from that particular state. Since in general the agents visit the same state of an SG multiple times, the stage game is a repeated game.

In a static or repeated game, the policy loses the state argument and transforms into a strategy σ_i: $U_i \to [0, 1]$. An agent's strategy for the stage game arising in some state of the SG is its policy for that state. MARL algorithms relying on the stagewise approach learn strategies separately for every stage game. The agent's overall policy is, then, the aggregate of these strategies.

Stochastic strategies (and consequently, stochastic policies) are of a more immediate importance in MARL than in single-agent RL, because in certain cases, like for the Nash equilibrium described later, the solutions can only be expressed in terms of stochastic strategies.

An important solution concept for static games, which will be used often in the sequel, is the Nash equilibrium. First, define the best response of agent i to a vector of opponent strategies as the strategy σ_i^* that achieves the maximum expected reward given these opponent strategies

$$E\{r_i \mid \sigma_1, \ldots, \sigma_i, \ldots, \sigma_n\} \leq E\{r_i \mid \sigma_1, \ldots, \sigma_i^*, \ldots, \sigma_n\} \; \forall \sigma_i. \tag{6}$$

A Nash equilibrium is a joint strategy $[\sigma_1^*, \ldots, \sigma_n^*]^{\mathrm{T}}$ such that each individual strategy σ_i^* is a best response to the others (see e.g., [11]). The Nash equilibrium describes a *status quo*, where no agent can benefit by changing its strategy as long as all other agents keep their strategies constant. Any static game has at least one (possibly stochastic) Nash equilibrium; some static games have multiple Nash equilibria. Nash equilibria are used by many MARL algorithms reviewed in the sequel, either as a learning goal, or both as a learning goal and directly in the update rules.

III. Benefits and Challenges in MARL

In addition to benefits owing to the distributed nature of the multiagent solution, such as the speedup made possible by parallel computation, multiple RL agents may harness new benefits from sharing experience, e.g., by communication, teaching, or imitation. Conversely, besides challenges inherited from single-agent RL, including the curse of dimensionality and the exploration–exploitation tradeoff, several new challenges arise in MARL: the difficulty of specifying a learning goal, the nonstationarity of the learning problem, and the need for coordination.

A. Benefits of MARL

A speedup of MARL can be realized thanks to parallel computation when the agents exploit the decentralized structure of the task. This direction has been investigated in, e.g., [45]–[50].

Experience sharing can help agents with similar tasks to learn faster and better. For instance, agents can exchange information using communication [51], skilled agents may serve as teachers for the learner [52], or the learner may watch and imitate the skilled agents [53].

When one or more agents fail in a multiagent system, the remaining agents can take over some of their tasks. This implies that MARL is inherently robust. Furthermore, by design, most multiagent systems also allow the easy insertion of new agents into the system, leading to a high degree of scalability.

Several existing MARL algorithms often require some additional preconditions to theoretically guarantee and to fully exploit the potential of these benefits [41], [53]. Relaxing these conditions and further improving the performance of the various MARL algorithms in this context is an active field of study.

[2]Full competition can also arise when more than two agents are involved. In this case, the reward functions must satisfy $\rho_1(x, \boldsymbol{u}, x') + \cdots + \rho_n(x, \boldsymbol{u}, x') = 0 \; \forall x, x' \in X, \boldsymbol{u} \in \boldsymbol{U}$. However, the literature on RL in fully competitive games typically deals with the two-agent case only.

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.

B. Challenges in MARL

The curse of dimensionality encompasses the exponential growth of the discrete state-action space in the number of state and action variables (dimensions). Since basic RL algorithms, like *Q-learning*, estimate values for each possible discrete state or state-action pair, this growth leads directly to an exponential increase of their computational complexity. The complexity of MARL is exponential also in the number of agents, because each agent adds its own variables to the joint state-action space.

Specifying a good MARL goal in the general stochastic game is a difficult challenge, as the agents' returns are correlated and cannot be maximized independently. Several types of MARL goals have been proposed in the literature, which consider stability of the agent's learning dynamics [54], adaptation to the changing behavior of the other agents [55], or both [13], [38], [56]–[58]. A detailed analysis of this open problem is given in Section IV.

Nonstationarity of the multiagent learning problem arises because all the agents in the system are learning simultaneously. Each agent is, therefore, faced with a moving-target learning problem: the best policy changes as the other agents' policies change.

The exploration–exploitation tradeoff requires online (single- as well as multiagent) RL algorithms to strike a balance between the exploitation of the agent's current knowledge, and exploratory, information-gathering actions taken to improve that knowledge. The ε-greedy policy (Section II-A) is a simple example of such a balance. The exploration strategy is crucial for the efficiency of RL algorithms. In MARL, further complications arise due to the presence of multiple agents. Agents explore to obtain information not only about the environment, but also about the other agents (e.g., for the purpose of building models of these agents). Too much exploration, however, can destabilize the learning dynamics of the other agents, thus making the learning task more difficult for the exploring agent.

The need for coordination stems from the fact that the effect of any agent's action on the environment depends also on the actions taken by the other agents. Hence, the agents' choices of actions must be mutually consistent in order to achieve their intended effect. Coordination typically boils down to consistently breaking ties between equally good actions or strategies. Although coordination is typically required in cooperative settings, it may also be desirable for self-interested agents, e.g., to simplify each agent's learning task by making the effects of its actions more predictable.

IV. MARL Goal

In fully cooperative SGs, the common return can be jointly maximized. In other cases, however, the agents' returns are different and correlated, and they cannot be maximized independently. Specifying a good MARL goal is, in general, a difficult problem.

In this section, the learning goals put forward in the literature are reviewed. These goals incorporate the *stability* of the learning dynamics of the agent on the one hand, and the *adaptation* to the dynamic behavior of the other agents on the other hand. Stability essentially means the convergence to a stationary policy, whereas adaptation ensures that performance is maintained or improved as the other agents are changing their policies.

The goals typically formulate conditions for static games, in terms of strategies and rewards. Some of the goals can be extended to dynamic games by requiring that the conditions are satisfied stagewise for all the states of the dynamic game. In this case, the goals are formulated in terms of stage strategies instead of strategies, and expected returns instead of rewards.

Convergence to equilibria is a basic stability requirement [42], [54]. It means the agents' strategies should eventually converge to a coordinated equilibrium. Nash equilibria are most frequently used. However, concerns have been voiced regarding their usefulness. For instance, in [14], it is argued that the link between stagewise convergence to Nash equilibria and performance in the dynamic SG is unclear.

In [13] and [56], convergence is required for stability, and rationality is added as an adaptation criterion. For an algorithm to be convergent, the authors of [13] and [56] require that the learner converges to a stationary strategy, given that the other agents use an algorithm from a predefined, targeted class of algorithms. Rationality is defined in [13] and [56] as the requirement that the agent converges to a best response when the other agents remain stationary. Though convergence to a Nash equilibrium is not explicitly required, it arises naturally if all the agents in the system are rational and convergent.

An alternative to rationality is the concept of no-regret, which is defined as the requirement that the agent achieves a return that is at least as good as the return of any stationary strategy, and this holds for any set of strategies of the other agents [57]. This requirement prevents the learner from "being exploited" by the other agents.

Targeted optimality/compatibility/safety are adaptation requirements expressed in the form of average reward bounds [55]. Targeted optimality demands an average reward, against a targeted set of algorithms, which is at least the average reward of a best response. Compatibility prescribes an average reward level in self-play, i.e., when the other agents use the learner's algorithm. Safety demands a safety-level average reward against all other algorithms. An algorithm satisfying these requirements does not necessarily converge to a stationary strategy.

Significant relationships of these requirements with other properties of learning algorithms discussed in the literature can be identified. For instance, opponent-independent learning is related to stability, whereas opponent-aware learning is related to adaptation [38], [59]. An opponent-independent algorithm converges to a strategy that is part of an equilibrium solution regardless of what the other agents are doing. An opponent-aware algorithm learns models of the other agents and reacts to them using some form of best response. Prediction and rationality, as defined in [58], are related to stability and adaptation, respectively. Prediction is the agent's capability to learn nearly accurate models of the other agents. An agent is called rational in [58] if it maximizes its expected return given its models of the other agents.

Table I summarizes these requirements and properties of the MARL algorithms. The stability and adaptation properties are

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.

TABLE I
STABILITY AND ADAPTATION IN MARL

Stability property	Adaptation property	Some relevant work
convergence	rationality	[13], [60]
convergence	no-regret	[57]
—	targeted optimality, compatibility, safety	[14], [55]
opponent-independent	opponent-aware	[38], [59]
equilibrium learning	best-response learning	[61]
prediction	rationality	[58]

given in the first two columns. Pointers to some relevant literature are provided in the last column.

Remarks: Stability of the learning process is needed, because the behavior of stable agents is more amenable to analysis and meaningful performance guarantees. Moreover, a stable agent reduces the nonstationarity in the learning problem of the other agents, making it easier to solve. Adaptation to the other agents is needed because their dynamics are generally unpredictable. Therefore, a good MARL goal must include both components. Since "perfect" stability and adaptation cannot be achieved simultaneously, an algorithm should guarantee bounds on both stability and adaptation measures. From a practical viewpoint, a realistic learning goal should also include bounds on the transient performance, in addition to the usual asymptotic requirements.

Convergence and rationality have been used in dynamic games in the stagewise fashion explained in the beginning of Section IV, although their extension to dynamic games was not explained in the papers that introduced them [13], [56]. No-regret has not been used in dynamic games, but it could be extended in a similar way. It is unclear how targeted optimality, compatibility, and safety could be extended.

V. TAXONOMY OF MARL ALGORITHMS

MARL algorithms can be classified along several dimensions, among which some, such as the task type, stem from properties of multiagent systems in general. Others, like awareness of the other agents, are specific to learning multiagent systems.

The type of task targeted by the learning algorithm leads to a corresponding classification of MARL techniques into those addressing fully cooperative, fully competitive, or mixed SGs. A significant number of algorithms are designed for static (stateless) tasks only. Fig. 1 summarizes the breakdown of MARL algorithms by task type.

The degree of awareness of other learning agents exhibited by MARL algorithms is strongly related to the targeted learning goal. Algorithms focused on stability (convergence) only are typically unaware and *independent* of the other learning agents. Algorithms that consider adaptation to the other agents clearly need to be *aware* to some extent of their behavior. If adaptation is taken to the extreme and stability concerns are disregarded, algorithms are only *tracking* the behavior of the other agents. The degree of agent awareness exhibited by the algorithms can be determined even if they do not explicitly target stability or adaptation goals. All agent-tracking algorithms and many agent-aware algorithms use some form of opponent modeling to keep track of the other agents' policies [40], [76], [77].

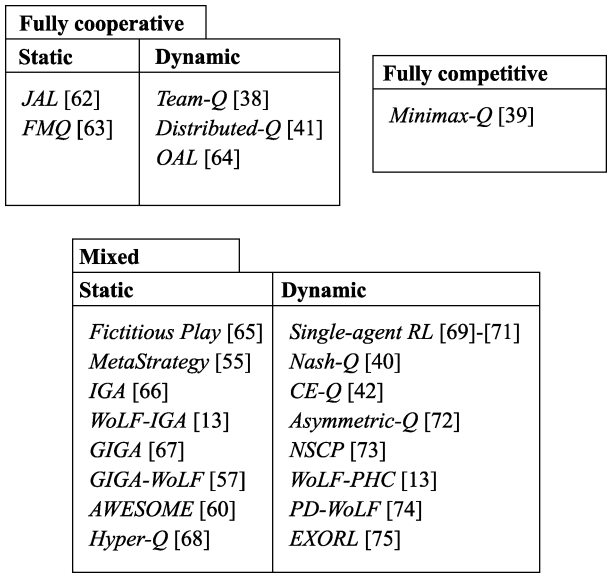


Fig. 1. Breakdown of MARL algorithms by the type of task they address.

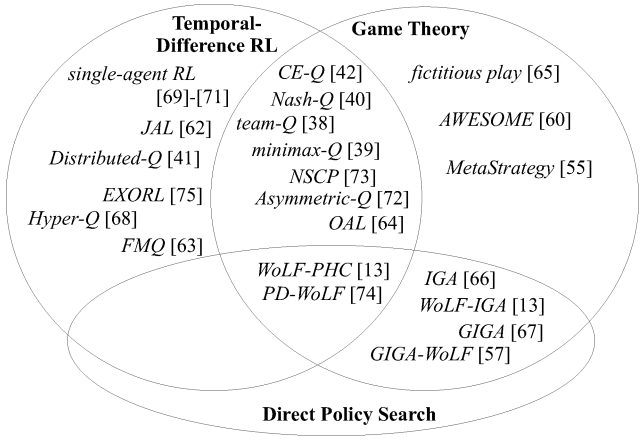


Fig. 2. MARL encompasses temporal-difference reinforcement learning, game theory, and direct policy search techniques.

The field of origin of the algorithms is a taxonomy axis that shows the variety of research inspiration benefiting MARL. MARL can be regarded as a fusion of temporal-difference RL, game theory, and more general, direct policy search techniques. Temporal-difference RL techniques rely on Bellman's equation and originate in dynamic programming. An example is the *Q-learning* algorithm. Fig. 2 presents the organization of the algorithms by their field of origin.

Other taxonomy axes include the following.[3]

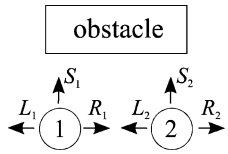
1) Homogeneity of the agents' learning algorithms: the algorithm only works if all the agents use it (homogeneous learning agents, e.g., *team-Q*, *Nash-Q*), or other agents can use other learning algorithms (heterogeneous learning agents, e.g., *AWESOME*, *WoLF-PHC*).

[3]All the mentioned algorithms are discussed separately in Section VI, where references are given for each of them.

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.

TABLE II
BREAKDOWN OF MARL ALGORITHMS BY TASK TYPE AND DEGREE OF AGENT AWARENESS

Task type → ↓ Agent awareness	Cooperative	Competitive	Mixed
Independent	coordination-free	opponent-independent	agent-independent
Tracking	coordination-based	—	agent-tracking
Aware	indirect coordination	opponent-aware	agent-aware



Q	L_2	S_2	R_2
L_1	10	-5	0
S_1	-5	-10	-5
R_1	-10	-5	10

Fig. 3. (Left) Two mobile agents approaching an obstacle need to coordinate their action selection. (Right) The common Q-values of the agents for the state depicted to the left.

2) Assumptions on the agent's prior knowledge of the task: a task model is available to the learning agent (model-based learning, e.g., *AWESOME*) or not (model-free learning, e.g., *team-Q*, *Nash-Q*, *WoLF-PHC*).
3) Assumptions on the agent's inputs. Typically, the inputs are assumed to exactly represent the state of the environment. Differences appear in the agent's observations of other agents: an agent might need to observe the actions of the other agents (e.g., *team-Q*, *AWESOME*), their actions and rewards (e.g., *Nash-Q*), or neither (e.g., *WoLF-PHC*).

VI. MARL ALGORITHMS

This section reviews a representative selection of algorithms that provides insight into the MARL state of the art. The algorithms are grouped first by the type of task addressed, and then by the degree of agent awareness, as depicted in Table II. Therefore, algorithms for fully cooperative tasks are presented first, in Section VI-A. Explicit coordination techniques that can be applied to algorithms in any class are discussed separately in Section VI-B. Algorithms for fully competitive tasks are reviewed in Section VI-C. Finally, Section VI-D presents algorithms for mixed tasks.

Algorithms that are designed only for static tasks are given separate paragraphs in the text. Simple examples are provided to illustrate several central issues that arise.

A. Fully Cooperative Tasks

In a fully cooperative SG, the agents have the same reward function ($\rho_1 = \cdots = \rho_n$) and the learning goal is to maximize the common discounted return. If a centralized controller were available, the task would reduce to a Markov decision process, the action space of which would be the joint action space of the SG. In this case, the goal could be achieved by learning the optimal joint-action values with *Q-learning*

$$Q_{k+1}(x_k, \boldsymbol{u}_k) = Q_k(x_k, \boldsymbol{u}_k) + \alpha\left[r_{k+1} + \gamma \max_{\boldsymbol{u}'} Q_k(x_{k+1}, \boldsymbol{u}') - Q_k(x_k, \boldsymbol{u}_k)\right] \quad (7)$$

and using the greedy policy. However, the agents are independent decision makers, and a coordination problem arises even if all the agents learn in parallel the common optimal Q-function using (7). It might seem that the agents could use greedy policies applied to Q^* to maximize the common return

$$\bar{h}_i^*(x) = \arg\max_{u_i} \max_{u_1,\ldots,u_{i-1},u_{i+1},\ldots,u_n} Q^*(x, \boldsymbol{u}). \quad (8)$$

However, the greedy action selection mechanism breaks ties randomly, which means that in the absence of additional mechanisms, different agents may break ties in (8) in different ways, and the resulting joint action may be suboptimal.

Example 1: The need for coordination. Consider the situation illustrated in Fig. 3: Two mobile agents need to avoid an obstacle while maintaining formation (i.e., maintaining their relative positions). Each agent has three available actions: go straight (S_i), left (L_i), or right (R_i).

For a given state (position of the agents), the Q-function can be projected into the space of the joint agent actions. For the state represented in Fig. 3 (left), a possible projection is represented in the table on the right. This table describes a fully cooperative static (stage) game. The rows correspond to the actions of agent 1, the columns to the actions of agent 2. If both agents go left, or both go right, the obstacle is avoided while maintaining the formation: $Q(L_1, L_2) = Q(R_1, R_2) = 10$. If agent 1 goes left, and agent 2 goes right, the formation is broken: $Q(L_1, R_2) = 0$. In all other cases, collisions occur and the Q-values are negative.

Note the tie between the two optimal joint actions: (L_1, L_2) and (R_1, R_2). Without a coordination mechanism, agent 1 might assume that agent 2 will take action R_2, and therefore, it takes action R_1. Similarly, agent 2 might assume that agent 1 will take L_1, and consequently, takes L_2. The resulting joint action (R_1, L_2) is largely suboptimal, as the agents collide.

1) Coordination-Free Methods: The *Team Q-learning* algorithm [38] avoids the coordination problem by assuming that the optimal joint actions are unique (which is rarely the case). Then, if all the agents learn the common Q-function in parallel with (7), they can safely use (8) to select these optimal joint actions and maximize their return.

The *Distributed Q-learning* algorithm [41] solves the cooperative task without assuming coordination and with limited computation (its complexity is similar to that of single-agent *Q-learning*). However, the algorithm only works in deterministic problems. Each agent i maintains a local policy $\bar{h}_i(x)$, and a local Q-function $Q_i(x, u_i)$, depending only on its own action. The local Q-values are updated only when the update leads to an increase in the Q-value

$$Q_{i,k+1}(x_k, u_{i,k}) = \max\left\{Q_{i,k}(x_k, u_{i,k}), r_{k+1} + \gamma \max_{u_i} Q_{i,k}(x_{k+1}, u_i)\right\}. \quad (9)$$

This ensures that the local Q-value always captures the maximum of the joint-action Q-values: $Q_{i,k}(x, u_i) = \max_{u_1,\ldots,u_{i-1},u_{i+1},\ldots,u_n} Q_k(x, \boldsymbol{u})$ at all k, where

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.

$\boldsymbol{u} = [u_1, \ldots, u_n]^{\mathrm{T}}$ with u_i fixed. The local policy is updated only if the update leads to an improvement in the Q-values:

$$\bar{h}_{i,k+1}(x_k) = \begin{cases} u_{i,k} & \text{if } \max_{u_i} Q_{i,k+1}(x_k, u_i) \\ & \quad > \max_{u_i} Q_{i,k}(x_k, u_i) \\ \bar{h}_{i,k}(x_k) & \text{otherwise.} \end{cases} \quad (10)$$

This ensures that the joint policy $[\bar{h}_{1,k}, \ldots, \bar{h}_{n,k}]^{\mathrm{T}}$ is always optimal with respect to the global Q_k. Under the conditions that the reward function is positive and $Q_{i,0} = 0\ \forall i$, the local policies of the agents provably converge to an optimal joint policy.

2) Coordination-Based Methods: Coordination graphs [45] simplify coordination when the global Q-function can be additively decomposed into local Q-functions that only depend on the actions of a subset of agents. For instance, in an SG with four agents, the decomposition might be $Q(x, \boldsymbol{u}) = Q_1(x, u_1, u_2) + Q_2(x, u_1, u_3) + Q_3(x, u_3, u_4)$. The decomposition might be different for different states. Typically (like in this example), the local Q-functions have smaller dimensions than the global Q-function. Maximization of the joint Q-value is done by solving simpler, local maximizations in terms of the local value functions, and aggregating their solutions. Under certain conditions, coordinated selection of an optimal joint action is guaranteed [45], [46], [48].

In general, all the coordination techniques described in Section VI-B next can be applied to the fully cooperative MARL task. For instance, a framework to explicitly reason about possibly costly communication is the communicative multiagent team decision problem [78].

3) Indirect Coordination Methods: Indirect coordination methods bias action selection toward actions that are likely to result in good rewards or returns. This steers the agents toward coordinated action selections. The likelihood of good values is evaluated using, e.g., models of the other agents estimated by the learner, or statistics of the values observed in the past.

a) Static tasks: Joint Action Learners (JAL) learn joint-action values and employ empirical models of the other agents' strategies [62]. Agent i learns models for all the other agents $j \neq i$, using

$$\hat{\sigma}_j^i(u_j) = \frac{C_j^i(u_j)}{\sum_{\tilde{u}_j \in U_j} C_j^i(\tilde{u}_j)} \quad (11)$$

where $\hat{\sigma}_j^i$ is agent i's model of agent j's strategy and $C_j^i(u_j)$ counts the number of times agent i observed agent j taking action u_j. Several heuristics are proposed to increase the learner's Q-values for the actions with high likelihood of getting good rewards given the models [62].

The *Frequency Maximum Q-value (FMQ)* heuristic is based on the frequency with which actions yielded good rewards in the past [63]. Agent i uses Boltzmann action selection (5), plugging in modified Q-values $\tilde{Q}_i$ computed with the formula

$$\tilde{Q}_i(u_i) = Q_i(u_i) + \nu \frac{C_{\max}^i(u_i)}{C^i(u_i)} r_{\max}(u_i) \quad (12)$$

where $r_{\max}(u_i)$ is the maximum reward observed after taking action u_i, $C_{\max}^i(u_i)$ counts how many times this reward has been observed, $C^i(u_i)$ counts how many times u_i has been taken, and ν is a weighting factor. Compared to single-agent *Q-learning*, the only additional complexity comes from storing and updating these counters. However, the algorithm only works for deterministic tasks, where variance in the rewards resulting from the agent's actions can only be the result of the other agents' actions. In this case, increasing the Q-values of actions that produced good rewards in the past steers the agent toward coordination.

b) Dynamic tasks: In *Optimal Adaptive Learning (OAL)*, virtual games are constructed on top of each stage game of the SG [64]. In these virtual games, optimal joint actions are rewarded with 1, and the rest of the joint actions with 0. An algorithm is introduced that, by biasing the agent toward recently selected optimal actions, guarantees convergence to a coordinated optimal joint action for the virtual game, and therefore, to a coordinated joint action for the original stage game. Thus, *OAL* provably converges to optimal joint policies in any fully cooperative SG. It is the only currently known algorithm capable of achieving this. This, however, comes at the cost of increased complexity: each agent estimates empirically a model of the SG, virtual games for each stage game, models of the other agents, and an optimal value function for the SG.

4) Remarks and Open Issues: All the methods presented earlier rely on exact measurements of the state. Many of them also require exact measurements of the other agents' actions. This is most obvious for coordination-free methods: if at any point the perceptions of the agents differ, this may lead different agents to update their Q-functions differently, and the consistency of the Q-functions and policies can no longer be guaranteed. Communication might help relax these strict requirements, by providing a way for the agents to exchange interesting data (e.g., state measurements or portions of Q-tables) rather than rely on exact measurements to ensure consistency [51].

Most algorithms also suffer from the curse of dimensionality. *Distributed Q-learning* and *FMQ* are exceptions in the sense that their complexity is not exponential in the number of agents (but they only work in restricted settings).

B. Explicit Coordination Mechanisms

A general approach to solving the coordination problem is to make sure that ties are broken by all agents in the same way. This clearly requires that random action choices are somehow coordinated or negotiated. Mechanisms for doing so, based on social conventions, roles, and communication, are described next (mainly following the description of Vlassis [2]). The mechanisms here can be used for any type of task (cooperative, competitive, or mixed).

Both social conventions and roles restrict the action choices of the agents. An agent role restricts the set of actions available to that agent prior to action selection, as in, e.g., [79]. This means that some or all of the ties in (8) are prevented.

Social conventions encode *a priori* preferences toward certain joint actions, and help break ties during action selection. If properly designed, roles or social conventions eliminate ties completely. A simple social convention relies on a unique

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.

ordering of agents and actions [80]. These two orderings must be known to all agents. Combining them leads to a unique ordering of joint actions, and coordination is ensured if in (8) the first joint action in this ordering is selected by all the agents.

Communication can be used to negotiate action choices, either alone or in combination with the aforementioned techniques, as in [2] and [81]. When combined with the aforementioned techniques, communication can relax their assumptions and simplify their application. For instance, in social conventions, if only an ordering between agents is known, they can select actions in turn, in that order, and broadcast their selection to the remaining agents. This is sufficient to ensure coordination.

Learning coordination approaches have also been investigated, where the coordination structures are learned online, instead of being hardwired into the agents at inception. The agents learn social conventions in [80], role assignments in [82], and the structure of the coordination graph together with the local Q-functions in [83].

Example 2: Coordination using social conventions in a fully cooperative task. In the earlier Section VI-A (see Fig. 3), suppose the agents are ordered such that agent 1 $<$ agent 2 ($a < b$ means that a precedes b in the chosen ordering), and the actions of both the agents are ordered in the following way: $L_i < R_i < S_i$, $i \in \{1, 2\}$. To coordinate, the first agent in the ordering of the agents, agent 1, looks for an optimal joint action such that its action component is the first in the ordering of its actions: (L_1, L_2). It then selects its component of this joint action, L_1. As agent 2 knows the orderings, it can infer this decision, and appropriately selects L_2 in response. If agent 2 would still face a tie [e.g., if (L_1, L_2) and (L_1, S_2) were both optimal], it could break this tie by using the ordering of its own actions [which because $L_2 < S_2$ would also yield (L_1, L_2)].

If communication is available, only the ordering of the agents has to be known. Agent 1, the first in the ordering, chooses an action by breaking ties in some way between the optimal joint actions. Suppose it settles on (R_1, R_2), and therefore, selects R_1. It then communicates this selection to agent 2, which can select an appropriate response, namely the action R_2.

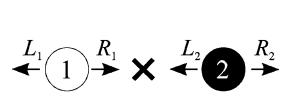
C. Fully Competitive Tasks

In a fully competitive SG (for two agents, when $\rho_1 = -\rho_2$), the minimax principle can be applied: maximize one's benefit under the worst-case assumption that the opponent will always endeavor to minimize it. This principle suggests using opponent-independent algorithms.

The *minimax-Q* algorithm [38], [39] employs the minimax principle to compute strategies and values for the stage games, and a temporal-difference rule similar to *Q-learning* to propagate the values across state-action pairs. The algorithm is given here for agent 1

$$h_{1,k}(x_k, \cdot) = \arg \mathbf{m}_1(Q_k, x_k) \tag{13}$$

$$\begin{aligned} Q_{k+1}(x_k, u_{1,k}, u_{2,k}) = {} & Q_k(x_k, u_{1,k}, u_{2,k}) \\ & + \alpha[r_{k+1} + \gamma\, \mathbf{m}_1(Q_k, x_{k+1}) \\ & - Q_k(x_k, u_{1,k}, u_{2,k})] \end{aligned} \tag{14}$$



Q	L_2	R_2
L_1	0	1
R_1	-10	10

Fig. 4. (Left) An agent (○) attempting to reach a goal (×) while avoiding capture by another agent (●). (Right) The Q-values of agent 1 for the state depicted to the left ($Q_2 = -Q_1$).

where $\mathbf{m}_1$ is the minimax return of agent 1

$$\mathbf{m}_1(Q, x) = \max_{h_1(x,\cdot)} \min_{u_2} \sum_{u_1} h_1(x, u_1) Q(x, u_1, u_2). \tag{15}$$

The stochastic strategy of agent 1 in state x at time k is denoted by $h_{1,k}(x, \cdot)$, with the dot standing for the action argument. The optimization problem in (15) can be solved by linear programming [84].

The Q-table is not subscripted by the agent index, because the equations make the implicit assumption that $Q = Q_1 = -Q_2$; this follows from $\rho_1 = -\rho_2$. *Minimax-Q* is truly opponent independent, because even if the minimax optimization has multiple solutions, any of them will achieve at least the minimax return regardless of what the opponent is doing.

If the opponent is suboptimal (i.e., does not always take the action that is most damaging the learner), and the learner has a model of the opponent's policy, it might actually do better than the minimax return (15). An opponent model can be learned using, e.g., the M^* algorithm described in [76], or a simple extension of (11) to multiple states

$$\hat{h}_j^i(x, u_j) = \frac{C_j^i(x, u_j)}{\sum_{\tilde{u}_j \in U_j} C_j^i(x, \tilde{u}_j)} \tag{16}$$

where $C_j^i(x, u_j)$ counts the number of times agent i observed agent j taking action u_j in state x.

Such an algorithm then becomes opponent aware. Even agent-aware algorithms for mixed tasks (see Section VI-D4) can be used to exploit a suboptimal opponent. For instance, *WoLF-PHC* was used with promising results on a fully competitive task in [13].

Example 3: The minimax principle. Consider the situation illustrated in the left part of Fig. 4: agent 1 has to reach the goal in the middle while still avoiding capture by its opponent, agent 2. Agent 2, on the other hand, has to prevent agent 1 from reaching the goal, preferably by capturing it. The agents can only move to the left or to the right.

For this situation (state), a possible projection of agent 1's Q-function onto the joint action space is given in the table on the right. This represents a zero-sum static game involving the two agents. If agent 1 moves left and agent 2 does likewise, agent 1 escapes capture, $Q_1(L_1, L_2) = 0$; furthermore, if at the same time, agent 2 moves right, the chances of capture decrease, $Q_1(L_1, R_2) = 1$. If agent 1 moves right and agent 2 moves left, agent 1 is captured, $Q_1(R_1, L_2) = -10$; however, if agent 2 happens to move right, agent 1 achieves the goal, $Q_1(R_1, R_2) = 10$. As agent 2's interests are opposite to those of agent 1, the Q-function of agent 2 is the negative of Q_1. For instance, when both agents move right, agent 1 reaches the goal and agent 3 is punished with a Q-value of -10.

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.

The minimax solution for agent 1 in this case is to move left, because for L_1, regardless of what agent 2 is doing, it can expect a return of at least 0, as opposed to -10 for R_1. Indeed, if agent 2 plays well, it will move left to protect the goal. However, it might *not* play well and move right instead. If this is true and agent 1 can find it out (e.g., by learning a model of agent 2), it can take advantage of this knowledge by moving right and achieving the goal.

D. Mixed Tasks

In mixed SGs, no constraints are imposed on the reward functions of the agents. This model is, of course, appropriate for self-interested agents, but even cooperating agents may encounter situations where their immediate interests are in conflict, e.g., when they need to compete for a resource. The influence of game-theoretic elements, like equilibrium concepts, is the strongest in the algorithms for mixed SGs. When multiple equilibria exist in a particular state of an SG, the equilibrium selection problem arises: the agents need to consistently pick their part of the same equilibrium.

A significant number of algorithms in this category are designed only for static tasks (i.e., repeated, general-sum games). In repeated games, one of the essential properties of RL, delayed reward, is lost. However, the learning problem is still nonstationary due to the dynamic behavior of the agents that play the repeated game. This is why most methods in this category focus on adaptation to other agents.

Besides agent-independent, agent-tracking, and agent-aware techniques, the application of single-agent RL methods to the MARL task is also presented here. That is because single-agent RL methods do not make any assumption on the type of task, and are therefore, applicable to mixed SGs, although without any guarantees for success.

1) Single-Agent RL: Single-agent RL algorithms like *Q-learning* can be directly applied to the multiagent case [69]. However, the nonstationarity of the MARL problem invalidates most of the single-agent RL theoretical guarantees. Despite its limitations, this approach has found a significant number of applications, mainly because of its simplicity [70], [71], [85], [86].

One important step forward in understanding how single-agent RL works in multiagent tasks was made recently in [87]. The authors applied results in evolutionary game theory to analyze the dynamic behavior of *Q-learning* with Boltzmann policies (5) in repeated games. It appeared that for certain parameter settings, *Q-learning* is able to converge to a coordinated equilibrium in particular games. In other cases, unfortunately, it seems that Q-learners may exhibit cyclic behavior.

2) Agent-Independent Methods: Algorithms that are independent of the other agents share a common structure based on *Q-learning*, where policies and state values are computed with game-theoretic solvers for the stage games arising in the states of the SG [42], [61]. This is similar to (13) and (14); the only difference is that for mixed games, solvers can be different from minimax.

Denoting by $\{Q_{\cdot,k}(x,\cdot)\}$ the stage game arising in state x and given by all the agents' Q-functions at time k, learning takes place according to

$$h_{i,k}(x,\cdot) = \mathbf{solve}_i\{Q_{\cdot,k}(x_k,\cdot)\} \tag{17}$$

$$\begin{aligned} Q_{i,k+1}(x_k,\boldsymbol{u}_k) = {} & Q_{i,k}(x_k,\boldsymbol{u}_k) \\ & + \alpha\big[r_{i,k+1} + \gamma\cdot\mathbf{eval}_i\{Q_{\cdot,k}(x_{k+1},\cdot)\} \\ & - Q_{i,k}(x_k,\boldsymbol{u}_k)\big] \end{aligned} \tag{18}$$

where $\mathbf{solve}_i$ returns agent i's part of some type of equilibrium (a strategy), and $\mathbf{eval}_i$ gives the agent's expected return given this equilibrium. The goal is the convergence to an equilibrium in every state.

The updates use the Q-tables of all the agents. So, each agent needs to replicate the Q-tables of the other agents. It can do that by applying (18). This requires two assumptions: that all agents use the same algorithm, and that all actions and rewards are exactly measurable. Even under these assumptions, the updates (18) are only guaranteed to maintain identical results for all the agents if $\mathbf{solve}$ returns consistent equilibrium strategies for all agents. This means the equilibrium selection problem arises when the solution of $\mathbf{solve}$ is not unique.

A particular instance of $\mathbf{solve}$ and $\mathbf{eval}$ for, e.g., *Nash Q-learning* [40], [54] is

$$\begin{cases} \mathbf{eval}_i\{Q_{\cdot,k}(x,\cdot)\} = V_i(x,\mathrm{NE}\{Q_{\cdot,k}(x,\cdot)\}) \\ \mathbf{solve}_i\{Q_{\cdot,k}(x,\cdot)\} = \mathrm{NE}_i\{Q_{\cdot,k}(x,\cdot)\} \end{cases} \tag{19}$$

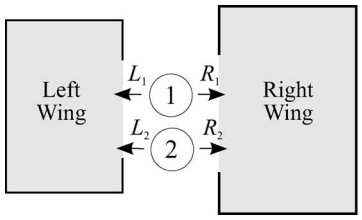
where NE computes a Nash equilibrium (a set of strategies), NE_i is agent i's strategy component of this equilibrium, and $V_i(x,\mathrm{NE}\{Q_{\cdot,k}(x,\cdot)\})$ is the expected return for agent i from x under this equilibrium. The algorithm provably converges to Nash equilibria for all states if either: 1) every stage game encountered by the agents during learning has a Nash equilibrium under which the expected return of all the agents is maximal or 2) every stage game has a Nash equilibrium that is a saddle point, i.e., not only does the learner not benefit from deviating from this equilibrium, but the other agents do benefit from this [40], [88]. This requirement is satisfied only in a small class of problems. In all other cases, some external mechanism for equilibrium selection is needed for convergence.

Instantiations of *correlated equilibrium Q-learning (CE-Q)* [42] or *asymmetric Q-learning* [72] can be performed in a similar fashion, by using correlated or Stackelberg (leader–follower) equilibria, respectively. For *asymmetric-Q*, the follower does not need to model the leader's Q-table; however, the leader must know how the follower chooses its actions.

Example 4: The equilibrium selection problem. Consider the situation illustrated in Fig. 5, left: Two cleaning robots (the agents) have arrived at a junction in a building, and each needs to decide which of the two wings of the building it will clean. It is inefficient if both agents clean the same wing, and both agents prefer to clean the left wing because it is smaller, and therefore, requires less time and energy.

For this situation (state), possible projections of the agents' Q-functions onto the joint action space are given in the tables on the right. These tables represent a general-sum static game involving

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.



Q_1	L_2	R_2
L_1	0	3
R_1	2	0

Q_2	L_2	R_2
L_1	0	2
R_1	3	0

Fig. 5. (Left) Two cleaning robots negotiating their assignment to different wings of a building. Both robots prefer to clean the smaller left wing. (Right) The Q-values of the two robots for the state depicted to the left.

the two agents. If both agents choose the same wing, they will not clean the building efficiently, $Q_1(L_1, L_2) = Q_1(R_1, R_2) = Q_2(L_1, L_2) = Q_2(R_1, R_2) = 0$. If agent 1 takes the (preferred) left wing and agent 2 the right wing, $Q_1(L_1, R_2) = 3$, and $Q_2(L_1, R_2) = 2$. If they choose the other way around, $Q_1(R_1, L_2) = 2$, and $Q_2(R_1, L_2) = 3$.

For these returns, there are two deterministic Nash equilibria[4]: (L_1, R_2) and (R_1, L_2). This is easy to see: if either agent unilaterally deviates from these joint actions, it can expect a (bad) return of 0. If the agents break the tie between these two equilibria independently, they might do so inconsistently and arrive at a suboptimal joint action. This is the equilibrium selection problem, corresponding to the coordination problem in fully cooperative tasks. Its solution requires additional coordination mechanisms, e.g., social conventions.

3) Agent-Tracking Methods: Agent-tracking algorithms estimate models of the other agents' strategies or policies (depending on whether static or dynamic games are considered) and act using some form of best-response to these models. Convergence to stationary strategies is not a requirement. Each agent is assumed capable to observe the other agents' actions.

a) Static tasks: In the *fictitious play* algorithm, agent i acts at each iteration according to a best response (6) to the models $\hat{\sigma}_1^i, \ldots, \hat{\sigma}_{i-1}^i, \hat{\sigma}_{i+1}^i, \ldots, \hat{\sigma}_n^i$ [65]. The models are computed empirically using (11). Fictitious play converges to a Nash equilibrium in certain restricted classes of games, among which are fully cooperative, repeated games [62].

The *MetaStrategy* algorithm, introduced in [55], combines modified versions of fictitious play, minimax, and a game-theoretic strategy called Bully [89] to achieve the targeted optimality, compatibility, and safety goals (see Section IV).

To compute best responses, the *fictitious play* and *MetaStrategy* algorithms require a model of the static task, in the form of reward functions.

The *Hyper-Q* algorithm uses the other agents' models as a state vector and learns a Q-function $Q_i(\hat{\sigma}_1, \ldots, \hat{\sigma}_{i-1}, \hat{\sigma}_{i+1}, \ldots, \hat{\sigma}_n, u_i)$ with an update rule similar to *Q-learning* [68]. By learning values of strategies instead of only actions, *Hyper-Q* should be able to adapt better to nonstationary agents. One inherent difficulty is that the action selection probabilities in the models are continuous variables. This means the classical, discrete-state *Q-learning* algorithm cannot be used. Less understood, approximate versions of it are required instead.

b) Dynamic tasks: The *Nonstationary Converging Policies (NSCP)* algorithm [73] computes a best response to the models and uses it to estimate state values. This algorithm is very similar to (13) and (14) and (17) and (18); this time, the stage game solver gives a best response

$$h_{i,k}(x_k, \cdot) = \arg \mathbf{br}_i(Q_{i,k}, x_k) \tag{20}$$

$$Q_{i,k+1}(x_k, \boldsymbol{u}_k) = Q_k(x_k, \boldsymbol{u}_k) + \alpha[r_{i,k+1} + \gamma \mathbf{br}_i(Q_{i,k}, x_{k+1}) - Q_k(x_k, \boldsymbol{u}_k)] \tag{21}$$

where the best-response value operator $\mathbf{br}$ is implemented as

$$\mathbf{br}_i(Q_i, x) = \max_{h_i(x,\cdot)} \sum_{u_1,\ldots,u_n} h_i(x, u_i) \cdot Q_i(x, u_1, \ldots, u_n) \prod_{j=1, j\neq i}^{n} \hat{h}_j^i(x, u_j). \tag{22}$$

The empirical models $\hat{h}_j^i$ are learned using (16). In the computation of $\mathbf{br}$, the value of each joint action is weighted by the estimated probability of that action being selected, given the models of the other agents [the product term in (22)].

4) Agent-Aware Methods: Agent-aware algorithms target convergence, as well as adaptation to the other agents. Some algorithms provably converge for particular types of tasks (mostly static), others use heuristics for which convergence is not guaranteed.

a) Static tasks: The algorithms presented here assume the availability of a model of the static task, in the form of reward functions. The *AWESOME* algorithm [60] uses fictitious play, but monitors the other agents and, when it concludes that they are nonstationary, switches from the best response in fictitious play to a centrally precomputed Nash equilibrium (hence the name: Adapt When Everyone is Stationary, Otherwise Move to Equilibrium). In repeated games, *AWESOME* is provably rational and convergent [60] according to the definitions from [56] and [13] given in Section IV.

Some methods in the area of direct policy search use gradient update rules that guarantee convergence in specific classes of static games: *Infinitesimal Gradient Ascent (IGA)* [66], *Win-or-Learn-Fast IGA (WoLF-IGA)* [13], *Generalized IGA (GIGA)* [67], and *GIGA-WoLF* [57]. For instance, *IGA* and *WoLF-IGA* work in two-agent, two-action games, and use similar gradient update rules

$$\begin{cases} \alpha_{k+1} = \alpha_k + \delta_{1,k} \dfrac{\partial E\{r_1 \mid \alpha, \beta\}}{\partial \alpha} \\ \beta_{k+1} = \beta_k + \delta_{2,k} \dfrac{\partial E\{r_2 \mid \alpha, \beta\}}{\partial \beta}. \end{cases} \tag{23}$$

The strategies of the agents are sufficiently represented by the probability of selecting the first out of the two actions, α for agent 1 and β for agent 2. *IGA* uses constant gradient steps $\delta_{1,k} = \delta_{2,k} = \delta$, and the average reward of the policies converges to Nash rewards for an infinitesimal step size (i.e., when

[4]There is also a stochastic (mixed) Nash equilibrium, where each agent goes left with a probability $3/5$. This is because the strategies $\sigma_1(L_1) = 3/5, \sigma_1(R_1) = 2/5$ and $\sigma_2(L_2) = 3/5, \sigma_2(R_2) = 2/5$ are best responses to one another. The expected return of this equilibrium for both agents is $6/5$, worse than for any of the two deterministic equilibria.

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.

$\delta \to 0$). In *WoLF-IGA*, $\delta_{i,k}$ switches between a smaller value when agent i is winning, and a larger value when it is losing (hence the name, Win-or-Learn-Fast). *WoLF-IGA* is rational by the definition in Section IV, and convergent for infinitesimal step sizes [13] ($\delta_{i,k} \to 0$ when $k \to \infty$).

b) Dynamic tasks: Win-or-Learn-Fast Policy Hill-Climbing (WoLF-PHC) [13] is a heuristic algorithm that updates Q-functions with the *Q-learning* rule (4), and policies with a WoLF rule inspired from (23)

$$h_{i,k+1}(x_k, u_i) = h_{i,k}(x_k, u_i) + \begin{cases} \delta_{i,k} & \text{if } u_i = \arg\max_{\tilde{u}_i} Q_{i,k+1}(x_k, \tilde{u}_i) \\ -\dfrac{\delta_{i,k}}{|U_i| - 1} & \text{otherwise} \end{cases} \tag{24}$$

$$\delta_{i,k} = \begin{cases} \delta_{\text{win}} & \text{if winning} \\ \delta_{\text{lose}} & \text{if losing.} \end{cases} \tag{25}$$

The gradient step $\delta_{i,k}$ is larger when agent i is losing than when it is winning: $\delta_{\text{lose}} > \delta_{\text{win}}$. For instance, in [13], δ_{lose} is two to four times larger than δ_{win}. The rationale is that the agent should escape fast from losing situations, while adapting cautiously when it is winning, in order to encourage convergence. The win/lose criterion in (25) is based either on a comparison of an average policy with the current one, in the original version of *WoLF-PHC*, or on the second-order difference of policy elements, in *PD-WoLF* [74].

The *Extended Optimal Response (EXORL)* heuristic [75] applies a complementary idea in two-agent tasks: the policy update is biased in a way that minimizes the other agent's incentive to deviate from its current policy. Thus, convergence to a coordinated Nash equilibrium is expected.

5) Remarks and Open Issues: Static, repeated games represent a limited set of applications. Algorithms for static games provide valuable theoretical results; these results should however be extended to dynamic SGs in order to become interesting for more general classes of applications (e.g., *WoLF-PHC* [13] is such an extension). Most static game algorithms also assume the availability of an exact task model, which is rarely the case in practice. Versions of these algorithms that can work with imperfect and/or learned models would be interesting (e.g., *GIGA-WoLF* [57]). Many algorithms for mixed SGs suffer from the curse of dimensionality, and are sensitive to imperfect observations; the latter holds especially for agent-independent methods.

Game theory induces a bias toward static (stagewise) solutions in the dynamic case, as seen, e.g., in the agent-independent Q-learning template (17)–(18) and in the stagewise win/lose criteria in *WoLF* algorithms. However, the suitability of such stagewise solutions in the context of the dynamic task is currently unclear [14], [17].

One important research step is understanding the conditions under which single-agent RL works in mixed SGs, especially in light of the preference toward single-agent techniques in practice. This was pioneered by the analysis in [87].

VII. Application Domains

MARL has been applied to a variety of problem domains, mostly in simulation but also to some real-life tasks. Simulated domains dominate for two reasons. The first reason is that results in simpler domains are easier to understand and to use for gaining insight. The second reason is that in real life, scalability and robustness to imperfect observations are necessary, and few MARL algorithms exhibit these properties. In real-life applications, more direct derivations of single-agent RL (see Section VI-D1) are preferred [70], [85], [86], [90].

In this section, several representative application domains are reviewed: distributed control, multirobot teams, trading agents, and resource management.

A. Distributed Control

In distributed control, a set of autonomous, interacting controllers act in parallel on the same process. Distributed control is a meta-application for cooperative multiagent systems: any cooperative multiagent system is a distributed control system where the agents are the controllers, and their environment is the controlled process. For instance, in cooperative robotic teams, the control algorithms of the robots identify with the controllers, and the robots' environment together with their sensors and actuators identify with the process.

Particular distributed control domains where MARL is applied are process control [90], control of traffic signals [91], [92], and control of electrical power networks [93].

B. Robotic Teams

Robotic teams (also called multirobot systems) are the most popular application domain of MARL, encountered under the broadest range of variations. This is mainly because robotic teams are a very natural application of multiagent systems, but also because many MARL researchers are active in the robotics field. The robots' environment is a real or simulated spatial domain, most often having two dimensions. Robots use MARL to acquire a wide spectrum of skills, ranging from basic behaviors like navigation to complex behaviors like playing soccer.

In navigation, each robot has to find its way from a starting position to a fixed or changing goal position, while avoiding obstacles and harmful interference with other robots [13], [54].

Area sweeping involves navigation through the environment for one of several purposes: retrieval of objects, coverage of as much of the environment surface as possible, and exploration, where the robots have to bring into sensor range as much of the environment surface as possible [70], [85], [86].

Multitarget observation is an extension of the exploration task, where the robots have to maintain a group of moving targets within sensor range [94], [95].

Pursuit involves the capture of moving targets by the robotic team. In a popular variant, several "predator" robots have to capture a "prey" robot by converging on it [83], [96].

Object transportation requires the relocation of a set of objects into given final positions and configurations. The mass or size of some of the objects may exceed the transportation capabilities

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.

of one robot, thus requiring several robots to coordinate in order to bring about the objective [86].

Robot soccer is a popular, complex test-bed for MARL, that requires most of the skills enumerated earlier [4], [97]–[100]. For instance, intercepting the ball and leading it into the goal involve object retrieval and transportation skills, while the strategic placement of the players in the field is an advanced version of the coverage task.

C. Automated Trading

Software trading agents exchange goods on electronic markets on behalf of a company or a person, using mechanisms such as negotiations and auctions. For instance, the Trading Agent Competition is a simulated contest where the agents need to arrange travel packages by bidding for goods such as plane tickets and hotel bookings [101].

MARL approaches to this problem typically involve temporal-difference [34] or Q-learning agents, using approximate representations of the Q-functions to handle the large state space [102]–[105]. In some cases, cooperative agents represent the interest of a single company or individual, and merely fulfil different functions in the trading process, such as buying and selling [103], [104]. In other cases, self-interested agents interact in parallel with the market [102], [105], [106].

D. Resource Management

In resource management, the agents form a cooperative team, and they can be one of the following.

1) Managers of resources, as in [5]. Each agent manages one resource, and the agents learn how to best service requests in order to optimize a given performance measure.
2) Clients of resources, as in [107]. The agents learn how to best select resources such that a given performance measure is optimized.

A popular resource management domain is network routing [108]–[110]. Other examples include elevator scheduling [5] and load balancing [107]. Performance measures include average job processing times, minimum waiting time for resources, resource usage, and fairness in servicing clients.

E. Remarks

Though not an application domain *per se*, game-theoretic, stateless tasks are often used to test MARL approaches. Not only algorithms specifically designed for static games are tested on such tasks (e.g., *AWESOME* [60], *MetaStrategy* [55], *GIGA-WoLF* [57]), but also others that can, in principle, handle dynamic SGs (e.g., *EXORL* [75]).

As an avenue for future work, note that distributed control is poorly represented as an MARL application domain. This includes not only complex systems such as traffic, power, or sensor networks, but also simpler dynamic processes that have been successfully used to study single-agent RL (e.g., various types of pendulum systems).

VIII. Outlook

In the previous sections of this survey, the benefits and challenges of MARL have been reviewed, together with the approaches to address these challenges and exploit the benefits. Specific discussions have been provided for each particular subject. In this section, more general open issues are given, concerning the suitability of MARL algorithms in practice, the choice of the multiagent learning goal, and the study of the joint environment and learning dynamics.

A. Practical MARL

Most MARL algorithms are applied to small problems only, like static games and small grid worlds. As a consequence, these algorithms are unlikely to scale up to real-life multiagent problems, where the state and action spaces are large or even continuous. Few of them are able to deal with incomplete, uncertain observations. This situation can be explained by noting that scalability and uncertainty are also open problems in single-agent RL. Nevertheless, improving the suitability of MARL to problems of practical interest is an essential research step. Next, we describe several directions in which this research can proceed, and point to some pioneering work done along these directions. Such work mostly combines single-agent algorithms with heuristics to account for multiple agents.

Scalability is the central concern for MARL as it stands today. Most algorithms require explicit tabular storage of the agents' Q-functions and possibly of their policies. This limits the applicability of the algorithms to problems with a relatively small number of discrete states and actions. When the state and action spaces contain a large number of elements, tabular storage of the Q-function becomes impractical. Of particular interest is the case when states and possibly actions are continuous variables, making exact Q-functions impossible to store. In these cases, approximate solutions must be sought, e.g., by extending to multiple agents the work on approximate single-agent RL [111]–[122]. A fair number of approximate MARL algorithms have been proposed: for discrete, large state-action spaces, e.g., [123], for continuous states and discrete actions, e.g., [96], [98], and [124], and finally for continuous states and actions, e.g., [95], and [125]. Unfortunately, most of these algorithms only work in a narrow set of problems and are heuristic in nature. Significant advances in approximate MARL can be made if the wealth of theoretical results on single-agent approximate RL is put to use [112], [113], [115]–[119].

A complementary avenue for improving scalability is the discovery and exploitation of the decentralized, modular structure of the multiagent task [45], [48]–[50].

Providing *domain knowledge* to the agents can greatly help them in learning solutions to realistic tasks. In contrast, the large size of the state-action space and the delays in receiving informative rewards mean that MARL without any prior knowledge is very slow. Domain knowledge can be supplied in several forms. If approximate solutions are used, a good way to incorporate domain knowledge is to structure the approximator in a way that ensures high accuracy in important regions of the state-action space, e.g., close to the goal. Informative reward functions, also

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.

rewarding promising behaviors rather than only the achievement of the goal, could be provided to the agents [70], [86]. Humans or skilled agents could teach unskilled agents how to solve the task [126]. Shaping is a technique whereby the learning process starts by presenting the agents with simpler tasks, and progressively moves toward complex ones [127]. Preprogrammed reflex behaviors could be built into the agents [70], [86]. Knowledge about the task structure could be used to decompose it into subtasks, and learn a modular solution with, e.g., hierarchical RL [128]. Last, but not the least, if a (possibly incomplete) task model is available, this model could be used with model-based RL algorithms to initialize Q-functions to reasonable, rather than arbitrary, values.

Incomplete, uncertain state measurements could be handled with techniques related to partially observable Markov decision processes [129], as in [130] and [131].

B. Learning Goal

The issue of a suitable MARL goal for dynamic tasks with dynamic, learning agents, is a difficult open problem. MARL goals are typically formulated in terms of static games. Their extension to dynamic tasks, as discussed in Section IV, is not always clear or even possible. If an extension via stage games is possible, the relationship between the extended goals and performance in the dynamic task is not clear, and is to the authors' best knowledge not made explicit in the literature. This holds for stability requirements, like convergence to equilibria [42], [54], as well as for adaptation requirements, like rationality [13], [56].

Stability of the learning process is needed, because the behavior of stable agents is more amenable to analysis and meaningful performance guarantees. Adaptation to the other agents is needed because their dynamics are generally unpredictable. Therefore, a good multiagent learning goal must include both components. This means that MARL algorithms should neither be totally independent of the other agents, nor just track their behavior without concerns for convergence.

Moreover, from a practical viewpoint, a realistic learning goal should include bounds on the transient performance, in addition to the usual asymptotic requirements. Examples of such bounds include maximum time constraints for reaching a desired performance level, or a lower bound on instantaneous performance levels. Some steps in this direction have been taken in [55] and [57].

C. Joint Environment and Learning Dynamics

The stagewise application of game-theoretic techniques to solve dynamic multiagent tasks is a popular approach. It may, however, not be the most suitable, given that both the environment and the behavior of learning agents are generally dynamic processes. So far, game-theory-based analysis has only been applied to the learning dynamics of the agents [28], [87], [132], while the dynamics of the environment have not been explicitly considered. We expect that tools developed in the area of robust control will play an important role in the analysis of the learning process as a whole (i.e., interacting environment and learning dynamics). In addition, this framework can incorporate prior knowledge on bounds for imperfect observations, such as noise-corrupted variables.

IX. Conclusion

MARL is a young, but active and rapidly expanding field of research. It integrates results from single-agent reinforcement learning, game theory, and direct search in the space of behaviors. The promise of MARL is to provide a methodology and an array of algorithms enabling the design of agents that learn the solution to a nonlinear, stochastic task about which they possess limited or no prior knowledge.

In this survey, we have discussed in detail a representative set of MARL techniques for fully cooperative, fully competitive, and mixed tasks. Algorithms for dynamic tasks were analyzed more closely, but techniques for static tasks were investigated as well. A classification of MARL algorithms was given, and the different viewpoints on the central issue of the MARL learning goal were presented. We have provided an outlook synthesizing several of the main open issues in MARL, together with promising ways of addressing these issues. Additionally, we have reviewed the main challenges and benefits of MARL, as well as several representative problem domains where MARL techniques have been applied.

Many avenues for MARL are open at this point, and many research opportunities present themselves. In particular, control theory can contribute in addressing issues such as stability of learning dynamics and robustness against uncertainty in observations or the other agents' dynamics. In our view, significant progress in the field of multiagent learning can be achieved by a more intensive cross fertilization between the fields of machine learning, game theory, and control theory.

References

[1] G. Weiss, Ed., *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Cambridge, MA: MIT Press, 1999.

[2] N. Vlassis. (2003, Sep.) A concise introduction to multiagent systems and distributed AI, Fac. Sci. Univ. Amsterdam, Amsterdam, The Netherlands, Tech. Rep. [Online]. Available: http://www.science.uva.nl/∼vlassis/cimasdai/cimasdai.pdf

[3] H. V. D. Parunak, "Industrial and practical applications of DAI," in *Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence*, G. Weiss, Ed. Cambridge, MA: MIT Press, 1999, ch. 9, pp. 377–412.

[4] P. Stone and M. Veloso, "Multiagent systems: A survey from the machine learning perspective," *Auton. Robots*, vol. 8, no. 3, pp. 345–383, 2000.

[5] R. H. Crites and A. G. Barto, "Elevator group control using multiple reinforcement learning agents," *Mach. Learn.*, vol. 33, no. 2–3, pp. 235–262, 1998.

[6] S. Sen and G. Weiss, "Learning in multiagent systems," in *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, G. Weiss, Ed. Cambridge, MA: MIT Press, 1999, ch. 6, pp. 259–298.

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

[8] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.

[9] V. Cherkassky and F. Mulier, *Learning from Data*. New York: Wiley, 1998.

[10] T. J. Sejnowski and G. E. Hinton, Eds., *Unsupervised Learning: Foundations of Neural Computation*. Cambridge, MA: MIT Press, 1999.

[11] T. Başsar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, 2nd ed. SIAM Series in Classics in Applied Mathematics. London, U.K.: Academic, 1999.

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.

[12] D. Fudenberg and D. K. Levine, *The Theory of Learning in Games*. Cambridge, MA: MIT Press, 1998.

[13] M. Bowling and M. Veloso, "Multiagent learning using a variable learning rate," *Artif. Intell.*, vol. 136, no. 2, pp. 215–250, 2002.

[14] Y. Shoham, R. Powers, and T. Grenager. (2003, May). "Multi-agent reinforcement learning: A critical survey," Comput. Sci. Dept., Stanford Univ., Stanford, CA, Tech. Rep. [Online]. Available: http://multiagent.stanford.edu/papers/MALearning_ACriticalSurvey_2003_0516.pdf

[15] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. London, U.K.: Oxford Univ. Press, 1996.

[16] K. D. Jong, *Evolutionary Computation: A Unified Approach*. Cambridge, MA: MIT Press, 2005.

[17] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Auton. Agents Multi-Agent Syst.*, vol. 11, no. 3, pp. 387–434, Nov. 2005.

[18] M. A. Potter and K. A. D. Jong, "A cooperative coevolutionary approach to function optimization," in *Proc. 3rd Conf. Parallel Probl. Solving Nat. (PPSN-III)*, Jerusalem, Israel, Oct. 9–14, 1994, pp. 249–257.

[19] S. G. Ficici and J. B. Pollack, "A game-theoretic approach to the simple coevolutionary algorithm," in *Proc. 6th Int. Conf. Parallel Probl. Solving Nat. (PPSN-VI)*, Paris, France, Sep. 18–20, 2000, pp. 467–476.

[20] L. Panait, R. P. Wiegand, and S. Luke, "Improving coevolutionary search for optimal multiagent behaviors," in *Proc. 18th Int. Joint Conf. Artif. Intell. (IJCAI-03)*, Acapulco, Mexico, Aug. 9–15, pp. 653–660.

[21] T. Haynes, R. Wainwright, S. Sen, and D. Schoenefeld, "Strongly typed genetic programming in evolving cooperation strategies," in *Proc. 6th Int. Conf. Genet. Algorithms (ICGA-95)*, Pittsburgh, PA, Jul. 15–19, pp. 271–278.

[22] R. Salustowicz, M. Wiering, and J. Schmidhuber, "Learning team strategies: Soccer case studies," *Mach. Learn.*, vol. 33, no. 2–3, pp. 263–282, 1998.

[23] T. Miconi, "When evolving populations is better than coevolving individuals: The blind mice problem," in *Proc. 18th Int. Joint Conf. Artif. Intell. (IJCAI-03)*, Acapulco, Mexico, Aug. 9–15, pp. 647–652.

[24] V. Könönen, "Gradient based method for symmetric and asymmetric multiagent reinforcement learning," in *Proc. 4th Int. Conf. Intell. Data Eng. Autom. Learn. (IDEAL-03)*, Hong Kong, China, Mar. 21–23, pp. 68–75.

[25] F. Ho and M. Kamel, "Learning coordination strategies for cooperative multiagent systems," *Mach. Learn.*, vol. 33, no. 2–3, pp. 155–177, 1998.

[26] J. Schmidhuber, "A general method for incremental self-improvement and multi-agent learning," in *Evolutionary Computation: Theory and Applications*, X. Yao, Ed. Singapore: World Scientific, 1999, ch. 3, pp. 81–123.

[27] J. M. Smith, *Evolution and the Theory of Games*. Cambridge, U.K.: Cambridge Univ. Press, 1982.

[28] K. Tuyls and A. Nowé, "Evolutionary game theory and multi-agent reinforcement learning," *Knowl. Eng. Rev.*, vol. 20, no. 1, pp. 63–90, 2005.

[29] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 2, 2nd ed. Belmont, MA: Athena Scientific, 2001.

[30] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1, 3rd ed. Belmont, MA: Athena Scientific, 2005.

[31] M. L. Puterman, *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. New York: Wiley, 1994.

[32] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, pp. 279–292, 1992.

[33] J. Peng and R. J. Williams, "Incremental multi-step Q-learning," *Mach. Learn.*, vol. 22, no. 1–3, pp. 283–290, 1996.

[34] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, pp. 9–44, 1988.

[35] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-5, no. 5, pp. 843–846, Sep./Oct. 1983.

[36] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Proc. 7th Int. Conf. Mach. Learn. (ICML-90)*, Austin, TX, Jun. 21–23, pp. 216–224.

[37] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Mach. Learn.*, vol. 13, pp. 103–130, 1993.

[38] M. L. Littman, "Value-function reinforcement learning in Markov games," *J. Cogn. Syst. Res.*, vol. 2, no. 1, pp. 55–66, 2001.

[39] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. 11th Int. Conf. Mach. Learn. (ICML-94)*, New Brunswick, NJ, Jul. 10–13, pp. 157–163.

[40] J. Hu and M. P. Wellman, "Multiagent reinforcement learning: Theoretical framework and an algorithm," in *Proc. 15th Int. Conf. Mach. Learn. (ICML-98)*, Madison, WI, Jul. 24–27, pp. 242–250.

[41] M. Lauer and M. Riedmiller, "An algorithm for distributed reinforcement learning in cooperative multi-agent systems," in *Proc. 17th Int. Conf. Mach. Learn. (ICML-00)*, Stanford Univ., Stanford, CA, Jun. 29–Jul. 2, pp. 535–542.

[42] A. Greenwald and K. Hall, "Correlated-Q learning," in *Proc. 20th Int. Conf. Mach. Learn. (ICML-03)*, Washington, DC, Aug. 21–24, pp. 242–249.

[43] T. Jaakkola, M. I. Jordan, and S. P. Singh, "On the convergence of stochastic iterative dynamic programming algorithms," *Neural Comput.*, vol. 6, no. 6, pp. 1185–1201, 1994.

[44] J. N. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Mach. Learn.*, vol. 16, no. 1, pp. 185–202, 1994.

[45] C. Guestrin, M. G. Lagoudakis, and R. Parr, "Coordinated reinforcement learning," in *Proc. 19th Int. Conf. Mach. Learn. (ICML-02)*, Sydney, Australia, Jul. 8–12, pp. 227–234.

[46] J. R. Kok, M. T. J. Spaan, and N. Vlassis, "Non-communicative multi-robot coordination in dynamic environment," *Robot. Auton. Syst.*, vol. 50, no. 2–3, pp. 99–114, 2005.

[47] J. R. Kok and N. Vlassis, "Sparse cooperative Q-learning," in *Proc. 21st Int. Conf. Mach. Learn. (ICML-04)*, Banff, AB, Canada, Jul. 4–8, pp. 481–488.

[48] J. R. Kok and N. Vlassis, "Using the max-plus algorithm for multiagent decision making in coordination graphs," in *Robot Soccer World Cup IX (RoboCup 2005)*. Lecture Notes in Computer Science, vol. 4020, Osaka, Japan, Jul. 13–19, 2005.

[49] R. Fitch, B. Hengst, D. Suc, G. Calbert, and J. B. Scholz, "Structural abstraction experiments in reinforcement learning," in *Proc. 18th Aust. Joint Conf. Artif. Intell. (AI-05),* Lecture Notes in Computer Science, vol. 3809, Sydney, Australia, Dec. 5–9, pp. 164–175.

[50] L. Buşoniu, B. De Schutter, and R. Babuška, "Multiagent reinforcement learning with adaptive state focus," in *Proc. 17th Belgian–Dutch Conf. Artif. Intell. (BNAIC-05)*, Brussels, Belgium, Oct. 17–18, pp. 35–42.

[51] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. 10th Int. Conf. Mach. Learn. (ICML-93)*, Amherst, OH, Jun. 27–29, pp. 330–337.

[52] J. Clouse, "Learning from an automated training agent," presented at the Workshop Agents that Learn from Other Agents, 12th Int. Conf. Mach. Learn. (ICML-95), Tahoe City, CA, Jul. 9–12.

[53] B. Price and C. Boutilier, "Accelerating reinforcement learning through implicit imitation," *J. Artif. Intell. Res.*, vol. 19, pp. 569–629, 2003.

[54] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, pp. 1039–1069, 2003.

[55] R. Powers and Y. Shoham, "New criteria and a new algorithm for learning in multi-agent systems," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS-04)*, Vancouver, BC, Canada, Dec. 13–18, vol. 17, pp. 1089–1096.

[56] M. Bowling and M. Veloso, "Rational and convergent learning in stochastic games," in *Proc. 17th Int. Conf. Artif. Intell. (IJCAI-01)*, San Francisco, CA, Aug. 4–10, 2001, pp. 1021–1026.

[57] M. Bowling, "Convergence and no-regret in multiagent learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS-04)*, Vancouver, BC, Canada, Dec.13–18, vol. 17, pp. 209–216.

[58] G. Chalkiadakis. (2003, Mar.). Multiagent reinforcement learning: Stochastic games with multiple learning players, Dept. of Comput. Sci., Univ. Toronto. Toronto, ON, Canada, Tech. Rep. [Online]. Available: http://www.cs.toronto.edu/~gehalk/DepthReport/DepthReport.ps

[59] M. Bowling and M. Veloso. (2000, Oct.). "An analysis of stochastic game theory for multiagent reinforcement learning," Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. [Online]. Available: http://www.cs.ualberta.ca/ bowling/papers/00tr.pdf

[60] V. Conitzer and T. Sandholm, "AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents," in *Proc. 20th Int. Conf. Mach. Learn. (ICML-03)*, Washington, DC, Aug. 21–24, pp. 83–90.

[61] M. Bowling, "Multiagent learning in the presence of agents with limitations," Ph.D. dissertation, Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, May 2003.

[62] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," in *Proc. 15th Nat. Conf. Artif. Intell.*

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.

10th Conf. Innov. Appl. Artif. Intell. (AAAI/IAAI-98), Madison, WI, Jul. 26–30, pp. 746–752.

[63] S. Kapetanakis and D. Kudenko, "Reinforcement learning of coordination in cooperative multi-agent systems," in *Proc. 18th Nat. Conf. Artif. Intell. 14th Conf. Innov. Appl. Artif. Intell. (AAAI/IAAI-02)*, Menlo Park, CA, Jul. 28–Aug. 1, pp. 326–331.

[64] X. Wang and T. Sandholm, "Reinforcement learning to play an optimal Nash equilibrium in team Markov games," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS-02)*, Vancouver, BC, Canada, Dec. 9–14, vol. 15, pp. 1571–1578.

[65] G. W. Brown, "Iterative solutions of games by fictitious play," in *Activitiy Analysis of Production and Allocation*, T. C. Koopmans, Ed. New York: Wiley, 1951, ch. XXIV, pp. 374–376.

[66] S. Singh, M. Kearns, and Y. Mansour, "Nash convergence of gradient dynamics in general-sum games," in *Proc. 16th Conf. Uncertainty Artif. Intell. (UAI-00)*, San Francisco, CA, Jun. 30–Jul. 3, pp. 541–548.

[67] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *Proc. 20th Int. Conf. Mach. Learn. (ICML-03)*, Washington, DC, Aug. 21–24, pp. 928–936.

[68] G. Tesauro, "Extending Q-learning to general adaptive multi-agent systems," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS-03)*, Vancouver, BC, Canada, Dec. 8–13, vol. 16.

[69] S. Sen, M. Sekaran, and J. Hale, "Learning to coordinate without sharing information," in *Proc. 12th Nat. Conf. Artif. Intell. (AAAI-94)*, Seattle, WA, Jul. 31–Aug. 4, pp. 426–431.

[70] M. J. Matarić, "Reinforcement learning in the multi-robot domain," *Auton. Robots*, vol. 4, no. 1, pp. 73–83, 1997.

[71] R. H. Crites and A. G. Barto, "Improving elevator performance using reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS-95)*, Denver, CO, Nov. 27–30, 1996, vol. 8, pp. 1017–1023.

[72] V. Könönen, "Asymmetric multiagent reinforcement learning," in *Proc. IEEE/WIC Int. Conf. Intell. Agent Technol. (IAT-03)*, Halifax, NS, Canada, Oct. 13–17, pp. 336–342.

[73] M. Weinberg and J. S. Rosenschein, "Best-response multiagent learning in non-stationary environments," in *Proc. 3rd Int. Joint Conf. Auton. Agents Multiagent Syst. (AAMAS-04)*, New York, NY, Aug. 19–23, pp. 506–513.

[74] B. Banerjee and J. Peng, "Adaptive policy gradient in multiagent learning," in *Proc. 2nd Int. Joint Conf. Auton. Agents Multiagent Syst. (AAMAS-03)*, Melbourne, Australia, Jul. 14–18, pp. 686–692.

[75] N. Suematsu and A. Hayashi, "A multiagent reinforcement learning algorithm using extended optimal response," in *Proc. 1st Int. Joint Conf. Auton. Agents Multiagent Syst. (AAMAS-02)*, Bologna, Italy, Jul. 15–19, pp. 370–377.

[76] D. Carmel and S. Markovitch, "Opponent modeling in multi-agent systems," in *Adaptation and Learning in Multi-Agent Systems*, G. Weiss and S. Sen, Eds. New York: Springer-Verlag, 1996, ch. 3, pp. 40–52.

[77] W. T. Uther and M. Veloso. (1997, Apr.). Adversarial reinforcement learning, School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. [Onlilne]. Available: http://www.cs.cmu.edu/afs/cs/user/will/www/papers/Uther97a.ps

[78] D. V. Pynadath and M. Tambe, "The communicative multiagent team decision problem: Analyzing teamwork theories and models," *J. Artif. Intell. Res.*, vol. 16, pp. 389–423, 2002.

[79] M. T. J. Spaan, N. Vlassis, and F. C. A. Groen, "High level coordination of agents based on multiagent Markov decision processes with roles," in *Proc. Workshop Coop. Robot., 2002 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS-02)*, Lausanne, Switzerland, Oct. 1, pp. 66–73.

[80] C. Boutilier, "Planning, learning and coordination in multiagent decision processes," in *Proc. 6th Conf. Theor. Aspects Rationality Knowl. (TARK-96)*, De Zeeuwse Stromen, The Netherlands, Mar. 17–20, pp. 195–210.

[81] F. Fischer, M. Rovatsos, and G. Weiss, "Hierarchical reinforcement learning in communication-mediated multiagent coordination," in *Proc. 3rd Int. Joint Conf. Auton. Agents Multiagent Syst. (AAMAS-04)*, New York, Aug. 19–23, pp. 1334–1335.

[82] M. V. Nagendra Prasad, V. R. Lesser, and S. E. Lander, "Learning organizational roles for negotiated search in a multiagent system," *Int. J. Hum. Comput. Stud.*, vol. 48, no. 1, pp. 51–67, 1998.

[83] J. R. Kok, P. J. 't Hoen, B. Bakker, and N. Vlassis, "Utile coordination: Learning interdependencies among cooperative agents," in *Proc. IEEE Symp. Comput. Intell. Games (CIG-05)*, Colchester, U.K., Apr. 4–6, pp. 29–36.

[84] S. Nash and A. Sofer, *Linear and Nonlinear Programming*. New York: McGraw-Hill, 1996.

[85] M. J. Matarić, "Reward functions for accelerated learning," in *Proc. 11th Int. Conf. Mach. Learn. (ICML-94)*, New Brunswick, NJ, Jul. 10–13, pp. 181–189.

[86] M. J. Matarić, "Learning in multi-robot systems," in *Adaptation and Learning in Multi-Agent Systems*, G. Weiss and S. Sen, Eds. New York: Springer-Verlag, 1996, ch. 10, pp. 152–163.

[87] K. Tuyls, P. J. 't Hoen, and B. Vanschoenwinkel, "An evolutionary dynamical analysis of multi-agent learning in iterated games," *Auton. Agents Multi-Agent Syst.*, vol. 12, no. 1, pp. 115–153, 2006.

[88] M. Bowling, "Convergence problems of general-sum multiagent reinforcement learning," in *Proc. 17th Int. Conf. Mach. Learn. (ICML-00)*, Stanford Univ., Stanford, CA, Jun. 29–Jul. 2, pp. 89–94.

[89] M. L. Littman and P. Stone, "Implicit negotiation in repeated games," in *Proc. 8th Int. Workshop Agent Theories Arch. Lang. (ATAL-2001)*, Seattle, WA, Aug. 21–24, pp. 96–105.

[90] V. Stephan, K. Debes, H.-M. Gross, F. Wintrich, and H. Wintrich, "A reinforcement learning based neural multi-agent-system for control of a combustion process," in *Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Netw. (IJCNN-00)*, Como, Italy, Jul. 24–27, pp. 6217–6222.

[91] M. Wiering, "Multi-agent reinforcement learning for traffic light control," in *Proc. 17th Int. Conf. Mach. Learn. (ICML-00)*, Stanford Univ., Stanford, CA, Jun. 29–Jul. 2, pp. 1151–1158.

[92] B. Bakker, M. Steingrover, R. Schouten, E. Nijhuis, and L. Kester, "Cooperative multi-agent reinforcement learning of traffic lights," presented at the Workshop Coop. Multi-Agent Learn., 16th Eur. Conf. Mach. Learn. (ECML-05), Porto, Portugal, Oct. 3.

[93] M. A. Riedmiller, A. W. Moore, and J. G. Schneider, "Reinforcement learning for cooperating and communicating reactive agents in electrical power grids," in *Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, M. Hannebauer, J. Wendler, and E. Pagello, Eds. New York: Springer, 2000, pp. 137–149.

[94] C. F. Touzet, "Robot awareness in cooperative mobile robot learning," *Auton. Robots*, vol. 8, no. 1, pp. 87–97, 2000.

[95] F. Fernández and L. E. Parker, "Learning in large cooperative multi-robot systems," *Int. J. Robot. Autom.*, vol. 16, no. 4, pp. 217–226, 2001.

[96] Y. Ishiwaka, T. Sato, and Y. Kakazu, "An approach to the pursuit problem on a heterogeneous multiagent system using reinforcement learning," *Robot. Auton. Syst.*, vol. 43, no. 4, pp. 245–256, 2003.

[97] P. Stone and M. Veloso, "Team-partitioned, opaque-transition reinforcement learning," in *Proc. 3rd Int. Conf. Auton. Agents (Agents-99)*, Seattle, WA, May 1–5, pp. 206–212.

[98] M. Wiering, R. Salustowicz, and J. Schmidhuber, "Reinforcement learning soccer teams with incomplete world models," *Auton. Robots*, vol. 7, no. 1, pp. 77–88, 1999.

[99] K. Tuyls, S. Maes, and B. Manderick, "Q-learning in simulated robotic soccer—large state spaces and incomplete information," in *Proc. 2002 Int. Conf. Mach. Learn. Appl. (ICMLA-02)*, Las Vegas, NV, Jun. 24–27, pp. 226–232.

[100] A. Merke and M. A. Riedmiller, "Karlsruhe brainstormers—A reinforcement learning approach to robotic soccer," in *Robot Soccer World Cup V (RoboCup 2001)*. Lecture Notes in Computer Science, vol. 2377, Washington, DC, Aug. 2–10, pp. 435–440.

[101] M. P. Wellman, A. R. Greenwald, P. Stone, and P. R. Wurman, "The 2001 trading agent competition," *Electron. Markets*, vol. 13, no. 1, pp. 4–12, 2003.

[102] W.-T. Hsu and V.-W. Soo, "Market performance of adaptive trading agents in synchronous double auctions," in *Proc. 4th Pacific Rim Int. Workshop Multi-Agents. Intell. Agents: Specification Model. Appl. (PRIMA-01)*. Lecture Notes in Computer Science Series, vol. 2132, Taipei, Taiwan, R.O.C., Jul. 28–29, pp. 108–121.

[103] J. W. Lee and J. Oo, "A multi-agent Q-learning framework for optimizing stock trading systems," in *Proc. 13th Int. Conf. Database Expert Syst. Appl. (DEXA-02)*. Lecture Notes in Computer Science, vol. 2453, Aix-en-Provence, France, Sep. 2–6, pp. 153–162.

[104] J. Oo, J. W. Lee, and B.-T. Zhang, "Stock trading system using reinforcement learning with cooperative agents," in *Proc. 19th Int. Conf. Mach. Learn. (ICML-02)*, Sydney, Australia, Jul. 8–12, pp. 451–458.

[105] G. Tesauro and J. O. Kephart, "Pricing in agent economies using multi-agent Q-learning," *Auton. Agents Multi-Agent Syst.*, vol. 5, no. 3, pp. 289–304, 2002.

[106] C. Raju, Y. Narahari, and K. Ravikumar, "Reinforcement learning applications in dynamic pricing of retail markets," in *Proc. 2003 IEEE Int. Conf. E-Commerce (CEC-03)*, Newport Beach, CA, Jun. 24–27, pp. 339–346.

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.

[107] A. Schaerf, Y. Shoham, and M. Tennenholtz, "Adaptive load balancing: A study in multi-agent learning," *J. Artif. Intell. Res.*, vol. 2, pp. 475–500, 1995.
[108] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS-93)*, Denver, CO, Nov. 29–Dec. 2, vol. 6, pp. 671–678.
[109] S. P. M. Choi and D.-Y. Yeung, "Predictive Q-routing: A memory-based reinforcement learning approach to adaptive traffic control," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS-95)*, Denver, CO, Nov. 27–30, vol. 8, pp. 945–951.
[110] P. Tillotson, Q. Wu, and P. Hughes, "Multi-agent learning for routing control within an Internet environment," *Eng. Appl. Artif. Intell.*, vol. 17, no. 2, pp. 179–185, 2004.
[111] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996.
[112] G. Gordon, "Stable function approximation in dynamic programming," in *Proc. 12th Int. Conf. Mach. Learn. (ICML-95)*, Tahoe City, CA, Jul. 9–12, pp. 261–268.
[113] J. N. Tsitsiklis and B. Van Roy, "Feature-based methods for large scale dynamic programming," *Mach. Learn.*, vol. 22, no. 1–3, pp. 59–94, 1996.
[114] R. Munos and A. Moore, "Variable-resolution discretization in optimal control," *Mach. Learn.*, vol. 49, no. 2–3, pp. 291–323, 2002.
[115] R. Munos, "Performance bounds in L_p-norm for approximate value iteration," *SIAM J. Control Optim.*, vol. 46, no. 2. pp. 546–561, 2007.
[116] C. Szepesvári and R. Munos, "Finite time bounds for sampling based fitted value iteration," in *Proc. 22nd Int. Conf. Mach. Learn. (ICML-05)*, Bonn, Germany, Aug. 7–11, pp. 880–887.
[117] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal difference learning with function approximation," *IEEE Trans. Autom. Control*, vol. 42, no. 5, pp. 674–690, May 1997.
[118] D. Ormoneit and S. Sen, "Kernel-based reinforcement learning," *Mach. Learn.*, vol. 49, no. 2–3, pp. 161–178, 2002.
[119] C. Szepesvári and W. D. Smart, "Interpolation-based Q-learning," in *Proc. 21st Int. Conf. Mach. Learn. (ICML-04)*, Banff, AB, Canada, Jul. 4–8.
[120] D. Ernst, P. Geurts, and L. Wehenkel, "Tree-based batch mode reinforcement learning," *J. Mach. Learn. Res.*, vol. 6, pp. 503–556, 2005.
[121] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *J. Mach. Learn. Res.*, vol. 4, pp. 1107–1149, 2003.
[122] S. Džeroski, L. D. Raedt, and K. Driessens, "Relational reinforcement learning," *Mach. Learn.*, vol. 43, no. 1–2, pp. 7–52, 2001.
[123] O. Abul, F. Polat, and R. Alhajj, "Multiagent reinforcement learning using function approximation," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 4, no. 4, pp. 485–497, Nov. 2000.
[124] L. Buşoniu, B. De Schutter, and R. Babuška, "Decentralized reinforcement learning control of a robotic manipulator," in *Proc. 9th Int. Conf. Control Autom. Robot. Vis. (ICARCV-06)*, Singapore, Dec. 5–8, pp. 1347–1352.
[125] H. Tamakoshi and S. Ishii, "Multiagent reinforcement learning applied to a chase problem in a continuous world," *Artif. Life Robot.*, vol. 5, no. 4, pp. 202–206, 2001.
[126] B. Price and C. Boutilier, "Implicit imitation in multiagent reinforcement learning," in *Proc. 16th Int. Conf. Mach. Learn. (ICML-99)*, Bled, Slovenia, Jun. 27–30, pp. 325–334.
[127] O. Buffet, A. Dutech, and F. Charpillet, "Shaping multi-agent systems with gradient reinforcement learning," *Auton. Agents Multi-Agent Syst.*, vol. 15, no. 2, pp. 197–220, 2007.
[128] M. Ghavamzadeh, S. Mahadevan, and R. Makar, "Hierarchical multi-agent reinforcement learning," *Auton. Agents Multi-Agent Syst.*, vol. 13, no. 2, pp. 197–229, 2006.
[129] W. S. Lovejoy, "Computationally feasible bounds for partially observed Markov decision processes," *Oper. Res.*, vol. 39, no. 1, pp. 162–175, 1991.
[130] S. Ishii, H. Fujita, M. Mitsutake, T. Yamazaki, J. Matsuda, and Y. Matsuno, "A reinforcement learning scheme for a partially-observable multi-agent game," *Mach. Learn.*, vol. 59, no. 1–2, pp. 31–54, 2005.
[131] E. A. Hansen, D. S. Bernstein, and S. Zilberstein, "Dynamic programming for partially observable stochastic games," in *Proc. 19th Natl. Conf. Artif. Intell. (AAAI-04)*, San Jose, CA, Jul. 25–29, pp. 709–715.
[132] J. M. Vidal, "Learning in multiagent systems: An introduction from a game-theoretic perspective," in *Adaptive Agents*. Lecture Notes in Artificial Intelligence, vol. 2636, E. Alonso, Ed. New York: Springer-Verlag, 2003, pp. 202–215.

Lucian Buşoniu received the M.Sc. degree and the Postgraduate Diploma from the Technical University of Cluj Napoca, Cluj Napoca, Romania, in 2003 and 2004, respectively, both in control engineering. Currently, he is working toward the Ph.D. degree at the Delft Center for Systems and Control, Faculty of Mechanical Engineering, Delft University of Technology, Delft, The Netherlands.

His current research interests include reinforcement learning in multiagent systems, approximate reinforcement learning, and adaptive and learning control.

Robert Babuška received the M.Sc. degree in control engineering from the Czech Technical University in Prague, Prague, Czech Republic, in 1990, and the Ph.D. degree from the Delft University of Technology, Delft, The Netherlands, in 1997.

He was with the Department of Technical Cybernetics, Czech Technical University in Prague and with the Faculty of Electrical Engineering, Delft University of Technology. Currently, he is a Professor at the Delft Center for Systems and Control, Faculty of Mechanical Engineering, Delft University of Technology. He is involved in several projects in mechatronics, robotics, and aerospace. He is the author or coauthor of more than 190 publications, including one research monograph (Kluwer Academic), two edited books, 24 invited chapters in books, 48 journal papers, and more than 120 conference papers. His current research interests include neural and fuzzy systems for modeling and identification, fault-tolerant control, learning, and adaptive control, and dynamic multiagent systems.

Prof. Babuška is the Chairman of the IFAC Technical Committee on Cognition and Control. He has served as an Associate Editor of the IEEE TRANSACTIONS ON FUZZY SYSTEMS, *Engineering Applications of Artificial Intelligence*, and as an Area Editor of *Fuzzy Sets and Systems*.

Bart De Schutter received the M.Sc. degree in electrotechnical-mechanical engineering and the Ph.D. degree in applied sciences (*summa cum laude* with congratulations of the examination jury) from Katholieke Universiteit Leuven (K.U. Leuven), Leuven, Belgium, in 1991 and 1996, respectively.

He was a Postdoctoral Researcher at the ESAT-SISTA Research Group of K.U. Leuven. In 1998, he was with the Control Systems Engineering Group, Faculty of Information Technology and Systems, Delft University of Technology, Delft, The Netherlands. Currently, he is a Full Professor at the Delft Center for Systems and Control, Faculty of Mechanical Engineering, Delft University of Technology, where he is also associated with the Department of Marine and Transport Technology. His current research interests include multiagent systems, intelligent transportation systems, control of transportation networks, hybrid systems control, discrete-event systems, and optimization.

Prof. De Schutter was the recipient of the 1998 SIAM Richard C. DiPrima Prize and the 1999 K.U. Leuven Robert Stock Prize for his Ph.D. thesis.

Authorized licensed use limited to: SAMRAT ASHOK TECHNOLOGICAL INSTITUTE. Downloaded on December 16, 2008 at 04:33 from IEEE Xplore. Restrictions apply.