Delft University of Technology

Delft Center for Systems and Control

Technical report 06-041

Reinforcement learning for multi-agent systems*

R. Babuška, L. Buşoniu, and B. De Schutter

July 2006

Paper for a keynote presentation at the *11th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2006)*, Prague, Czech Republic, Sept. 2006.

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: https://www.dcsc.tudelft.nl

* This report can also be downloaded via https://pub.bartdeschutter.org/abs/06_041.html

Reinforcement Learning for Multi-Agent Systems

Robert Babuška   Lucian Buşoniu   Bart De Schutter
Delft Center for Systems and Control, Delft University of Technology
Mekelweg 2, 2628 CD Delft, The Netherlands
r.babuska@tudelft.nl, i.l.busoniu@tudelft.nl, b.deschutter@tudelft.nl

Abstract

Multi-agent systems are rapidly finding applications in a variety of domains, including robotics, distributed control, telecommunications, etc. Although the individual agents can be programmed in advance, many tasks require that they learn behaviors online. A significant part of the research on multi-agent learning concerns reinforcement learning techniques. This paper gives a survey of multi-agent reinforcement learning, starting with a review of the different viewpoints on the learning goal, which is a central issue in the field. Two generic goals are distinguished: stability of the learning dynamics, and adaptation to the other agents' dynamic behavior. The focus on one of these goals, or a combination of both, leads to a categorization of the methods and approaches in the field. The challenges and benefits of multi-agent reinforcement learning are outlined along with open issues and future research directions.

1 Introduction

Multi-agent systems are rapidly finding applications in a wide variety of domains such as data mining, distributed control, collaborative decision support systems, robotic teams, etc. Although the individual agents can be programmed to exhibit some basic behaviors, many tasks require that agents learn new behaviors online, such that the performance of the agent or of the whole multi-agent system gradually improves.

Reinforcement learning (RL) agents learn by interacting with their environment, using only scalar rewards as feedback [1]. The simplicity and generality of this setting make it attractive also for multi-agent systems. However, the main challenge in multiagent RL (MARL) is that each learning agent must explicitly consider the other learning (and therefore, nonstationary) agents, and coordinate its behavior with theirs, such that a coherent joint behavior results.

Over the last years, many algorithms addressing this problem were proposed, fusing developments in the areas of temporal-difference RL , game theory and more general direct policy search techniques. MARL surveys typically review the field from a game-theoretic perspective [2–5]. The aim of our survey is to take a broader perspective and give an overall view of the field. We also address the different viewpoints on the learning goal in MARL , which led to certain diversity in the set of MARL algorithms and techniques. Finally, we identify open issues and further research directions, and outline control-theoretic ways to follow some of these directions.

This paper is organized as follows. Section 2 introduces the necessary background. Section 3 addresses the problem of a suitable multi-agent learning goal, and Section 4 reviews a representative selection of the literature on MARL , classifying algorithms by the type of task they solve. Section 5 concludes the paper.

2 Background

In single-agent RL, the environment (process) is described by a Markov decision process [1]. A definition of an MDP is given first, followed by its extension to the multi-agent case – the stochastic game.

2.1 The single-agent case

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

As a result of action u_k, the environment changes state from x_k to a next state x_{k+1} with probability $f(x_k, u_k, x_{k+1})$. The agent receives (possibly delayed) feedback on its performance via the scalar reward signal $r_{k+1} \in \mathbb{R}$, $r_{k+1} = \rho(x_k, u_k, x_{k+1})$. For deterministic models, the state transition distribution is replaced by a function $\overline{f} : X \times U \to X$. This means the reward only depends on the current state and action, $\overline{\rho} : X \times U \to \mathbb{R}$. The agent chooses actions according to its *policy* that may be either stochastic, $h : X \times U \to [0, 1]$, or deterministic, $\overline{h} : X \to U$. 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 discounted return:

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

where $\gamma \in (0, 1)$ is the discount factor. The *action-value function* (Q-function), $Q_i^h : X \times U \to \mathbb{R}$, is the expected return of a state-action pair under a given policy: $Q_i^h(x,u) = E\{R_k \,|\, x_k = x, u_k = u, h\}$. The agent can maximize its return by first computing the *optimal* Q-function, defined as $Q^*(x,u) = \max_h Q^h(x,u)$, and then choosing actions by the greedy policy $h^*(x) = \arg\max_u Q^*(x,u)$, which is optimal.

The *Q-learning* algorithm iteratively approximates Q^* by interaction with the environment, using observed rewards r_{k+1} and pairs of subsequent states x_k, x_{k+1} [6]:

$$Q_{k+1}(x_k,u_k) = Q_k(x_k,u_k) + \\ \alpha\big[r_{k+1} + \gamma \max_{u'} Q_k(x_{k+1},u') - Q_k(x_k,u_k)\big], \quad (2)$$

where $\alpha \in (0,1]$ is the learning rate. In [6] it is proved that the sequence Q_k converges to Q^* under certain conditions, including that the agent keeps trying all actions in all states with nonzero probability. This means that the agent must sometimes *explore*, i.e., perform other actions than dictated by the current greedy policy.

Example 1 *Path optimization through Q-learning.* Consider the following simple instance of a path-optimization problem. A single agent has to find by trial and error a shortest path from its starting position to some fixed goal (Figure 1). The agent receives a reward of -0.1 at each intermediate time step, and 10 upon reaching the goal. Then the trial terminates and the agent is reset to its initial position. The negative reward encodes 'energy consumption', and leads to the minimum-distance solution.

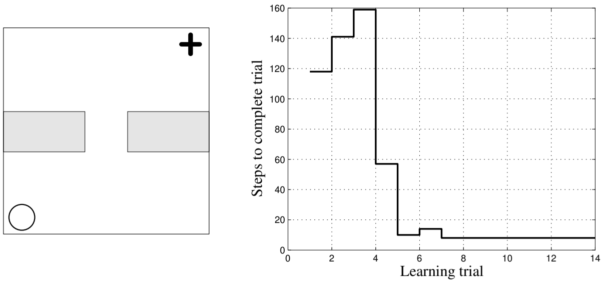


Figure 1. Left: a single-agent path-optimization problem: a robotic agent (circle) has to find the shortest path to the goal (cross), while avoiding obstacles (gray areas). Right: a typical convergence plot.

The state x is the agent's position, discretized on a 5×5 grid. The available actions u are the moves one cell further in one of the four compass directions, or standing still.

The minimum-time path is learned with Q-learning, using $\gamma = 0.98$ and a constant learning rate $\alpha = 0.2$. The agent explores by choosing at each step, with exponentially decaying probability, a random action instead of the greedy one. An eligibility trace with a decay factor $\lambda = 0.5$ is used to speed up the convergence [1]. Q-learning converges to an optimal policy in 7 trials. Note that convergence is not monotonous, mainly due to exploration: e.g., trial 6 takes longer than trial 5. □

2.2 The multi-agent case

The underlying model in the multi-agent case is the *stochastic game* [7].

Definition 2 *A stochastic game (SG) is a tuple* $\langle A, X, \{U_i\}_{i\in A}, f, \{\rho_i\}_{i\in A}\rangle$ *where:* $A = \{1,\ldots,n\}$ *is the set of* n *agents,* X *is the discrete set of environment states,* $\{U_i\}_{i\in A}$ *are the discrete sets of actions available to the agents, yielding the joint action set* $\boldsymbol{U} = \times_{i\in A} U_i$, $f : X \times \boldsymbol{U} \times X \to [0,1]$ *is the state transition probability distribution, and* $\rho_i : X \times \boldsymbol{U} \times X \to \mathbb{R}, i \in A$ *are the reward probability functions of the agents.*

Note that the state transitions, agent rewards $r_{i,k+1}$, and thus also the agent returns $R_{i,k}$, depend on the *joint action* $\boldsymbol{u}_k = [u_{1,k},\ldots,u_{n,k}]^{\mathrm{T}}$, $\boldsymbol{u}_k \in \boldsymbol{U}$, $u_{i,k} \in U_i$. 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 $X = \emptyset$, the SG reduces to a matrix game. A matrix game, when played repeatedly by the same agents, is called a repeated game. If $\rho_1 \equiv \cdots \equiv \rho_n$, the SG is fully cooperative. If $n = 2$ and $\rho_1 \equiv -\rho_2$, the SG is fully competitive.

In a matrix game, the policy loses the state argument and transforms into a strategy $h : U \to [0,1]$. Similarly, a policy conditioned on a given state x yields a strategy. The best response of agent i to a set of opponent strategies is a strategy that achieves the maximum expected reward given the opponents' strategies. A *Nash equilibrium* is a set of strategies such that each is a best-response to the others.

3 Multi-agent learning goal

In fully cooperative SG s, the common return can be jointly maximized. In other cases, however, specifying a good MARL goal is difficult, because the agents' returns are correlated and cannot be maximized independently.

In this section, we review the learning goals put forward in the literature. These goals incorporate *stability* of the learning process on the one hand, and *adaptation* to the dynamic behavior of the other agents on the other hand. Stability essentially means the convergence to stationary policies, whereas adaptation ensures that performance is maintained or improved.

Convergence to equilibria is a basic stability requirement, postulated already in the early MARL literature [8, 9]. Nash equilibria are most frequently used. However, concerns have been voiced regarding their usefulness [2], because of the unclear link to performance in dynamic tasks and their inherent conservatism.

Bowling and Veloso [7] add *rationality* as an adaptation criterion. Rationality means that the agent's policy converges to a best response when other agents remain stationary. An alternative to rationality is the concept *no-regret*, which prevents the learner from 'being exploited' by the other agents [10].

Targeted optimality / compatibility / safety [11] replace convergence with adaptation requirements, in the form of average reward bounds for three classes of opponents: those deemed interesting (targeted), those using the learner's algorithm, and remaining opponents.

Table 1 summarizes the desirable properties of MARL algorithms, as discussed above and in the literature. The focus on stability, adaptation, or a mix of the two, leads to a categorization of MARL algorithms into opponent-independent, opponent-tracking, and opponent-aware, as discussed in the next section.

Table 1. Stability and adaptation in multi-agent learning.

Stability property	Adaptation property	Ref.
convergence	rationality	[7,12]
convergence	no-regret	[10]
opponent-independent	opponent-aware	[5,13]
prediction	rationality	[4]
—	targeted optimality compatibility, safety	[2,11]

4 Multi-agent reinforcement learning algorithms

The MARL algorithms are organized here by the type of task they address: fully cooperative, fully competitive, and mixed tasks.

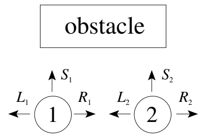
4.1 Fully cooperative tasks

In a fully cooperative SG , the reward functions are identical: $\rho_1 \equiv \ldots \equiv \rho_n$ and the learning goal is to maximize the common discounted return. If the agents are considered together as a centralized controller, the task reduces to a Markov decision process whose action space is the joint action space of the SG . The goal can be achieved by learning the optimal joint-action values with Q-learning:

$$Q_{k+1}(x_k, \boldsymbol{u}_k) = Q_k(x_k, \boldsymbol{u}_k) + \alpha\big[r_{k+1} + \gamma \max_{\boldsymbol{u}'} Q_k(x_{k+1}, \boldsymbol{u}') - Q_k(x_k, \boldsymbol{u}_k)\big], \quad (3)$$

and using the greedy policy. However, if the agents are independent decision makers, a coordination problem arises even if all the agents use the same learning algorithm. The greedy action selection mechanism breaks ties randomly, which means that in the absence of additional mechanisms, different agents may break a tie in different ways. The resulting joint action may be suboptimal.

Example 2 *Coordination in action selection.* Consider the situation illustrated in Figure 2: two mobile agents need to avoid an obstacle while maintaining formation.



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

Figure 2. Two mobile agents approaching an obstacle need to coordinate their action selection mechanism.

For the given state (positions of the agents), the Q-function can be projected into the space of the joint agent actions. An example of such a table is given in the right part of Figure 2. The rows correspond to actions of agent 1, the columns to 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. □

The following approaches to solving the coordination issue can be distinguished in the literature: coordination-free methods, direct and indirect coordination methods.

4.1.1 Coordination-free methods

The *Team Q-learning* algorithm [13] assumes that the optimal joint actions are unique (which will rarely be the case) and use (3) directly.

The *Distributed Q-learning* algorithm [14] solves the cooperative task without assuming coordination, however it is only valid in the deterministic setting (with $\alpha = 1$). Each agent i maintains an explicit policy $h_i(x)$, and a local Q-function $Q_i(x, u_i)$, depending only on its own action. Both are updated only in the direction that increases Q_i:

$$Q_{i,k+1}(x_k, u_{i,k}) = \max\big\{Q_{i,k}(x_k, u_{i,k}), r_{k+1} + \gamma \max_{u_i} Q_{i,k}(x_{k+1}, u_i)\big\} \quad (4)$$

$$h_{i,k+1}(x_k) = \begin{cases} u_{i,k} & \text{if } \max_{u_i} Q_{i,k+1}(x_k, u_i) \\ & \quad \neq \max_{u_i} Q_{i,k}(x_k, u_i) \\ h_{i,k}(x_k) & \text{otherwise} \end{cases} \quad (5)$$

Under the conditions that $Q_{i,0} \equiv 0$ and the common reward function is positive, the policies of the agents provably converge to the optimal joint policy $\boldsymbol{h}^*$.

4.1.2 Direct coordination methods

A more 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 the action choices are somehow coordinated or negotiated:

- *Social conventions* [15] and *roles* [16] restrict the action choices of the agents.
- *Coordination graphs* explicitly represent where coordination between agents is required, thus preventing the agents from engaging in unnecessary coordination activities [17].
- *Communication* is used to negotiate action choices, either alone or in combination with the above techniques, see, e.g., [18, 19].

4.1.3 Indirect coordination methods

In this class of approaches, action selection is biased toward actions that promise to yield better values, and thus steer the agents toward coordination. *Joint Action Learners* (JAL) [20] employ empirically learned models of the other agents' behavior. The *Frequency Maximum Q-value* (FMQ) heuristic [21] is based on the frequency with which actions yielded good values in the past. In *Optimal Adaptive Learning* (OAL), the bias is towards recently chosen optimal joint actions [22]. Using an additional mechanism to guarantee that these actions are eventually selected, OAL provably converges to optimal joint policies (at the cost of increased complexity).

Example 3 *Multi-agent coordination.* In this multi-agent coordination problem, two agents live in a grid world similar to that of Example 1. However, here they have to reach the goal cell simultaneously. This is represented by a reward of 10 upon simultaneously reaching the goal.

Coordination is needed around the goal (the agents need to move to it simultaneously), and near the passage in the middle of the grid-world (a collision would occur there if both agents took their own time-optimal paths, so one of them has to wait for the other one to pass first).

The algorithm used is team Q-learning with the addition of a simple social convention: an ordering of the joint actions which is known to both agents. The discount factor is $\gamma = 0.98$ and the learning rate at step k is $\alpha_k = \frac{1}{1+k/500}$. The agents explore by choosing at each step, with exponentially decaying probabilities, a random action instead of the greedy one. The agents converge to an optimal joint policy in 23 trials. Compared to the single-agent case in Figure 1, the agents initially take much longer to discover the goal (approximately 950 steps vs. 160 steps), and the convergence is much slower (21 trials vs. 7 trials). This is because of the relatively large state-action space: $(5 \cdot 5)^2$ states times 5^2 joint actions = 15265 state-action pairs, as opposed to 125 in the single-agent case. □

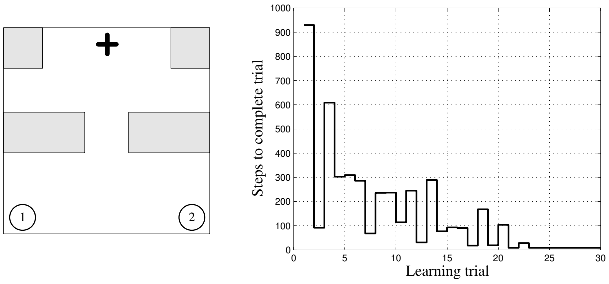


Figure 3. Left: a multi-agent coordination problem: two robotic agent (circles) have to reach the goal (cross) simultaneously, while avoiding obstacles (gray areas) and minimizing the distance traveled. Right: a typical convergence plot.

4.2 Fully competitive tasks

In a fully competitive SG (for two agents, $\rho_1 \equiv -\rho_2$), the minimax principle can be applied: maximize one's benefit under the assumption that the opponent will always act so as to minimize it. The resulting algorithm is *minimax-Q* [13], given here for agent 1:

$$h_{1,k}(x_k, \cdot) = \arg \mathbf{m}_1(Q_k, x_k) \tag{6}$$

$$\begin{aligned} Q_{k+1}(x_k, u_{1,k}, u_{2,k}) &= Q_k(x_k, u_{1,k}, u_{2,k}) + \\ \alpha\big[r_{k+1} + \gamma \mathbf{m}_1(Q_k, x_{k+1}) &- Q_k(x_k, u_{1,k}, u_{2,k})\big] \end{aligned} \tag{7}$$

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

$$\mathbf{m}_1(Q, x) = \max_{\tilde{h}_1(x,\cdot)} \min_{u_2} \sum_{u_1} \tilde{h}_1(x, u_1) Q(x, u_1, u_2) \tag{8}$$

The Q-table is not subscripted by the agent index, because the equations use the implicit assumption that $Q_2 = -Q_1 = -Q$. The coordination problem does not arise here, 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. Thus, minimax-Q is truly opponent-independent. However, if the learner has a model of the opponent's policy (i.e., is opponent-aware), it might actually do better than the minimax return (8).

4.3 Mixed tasks

In the general case, the reward functions of the agents may differ. This is certainly true 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 some resource. Mixed, dynamic tasks, represented by the unrestricted SG , exhibit all the MARL challenges: delayed reward, nonstationary opponents, and conflicting goals.

Single-agent RL can be directly applied to the multi-agent case [23]. However, the nonstationarity of the MARL problem invalidates most of the single-agent RL theoretical results. Therefore, single-agent RL will only work when agents do not severely interfere with one another. Despite its limitations, this approach found applications, mainly because of its simplicity [24,25]. In applications, information about other agents is typically encoded in the learner's input, thus indirectly enabling it to make decisions on the basis of their behavior.

Game-theoretic concepts of equilibria can be used to facilitate convergence. Algorithms in this category typically require that the agents know the task model (i.e., the reward function), and assume observable actions (some of them, even observable strategies).

In the sequel, we distinguish opponent-independent methods (targeting convergence), opponent-tracking methods (targeting rationality, i.e., the adaptation to opponents' strategies) and opponent-aware methods (targeting both convergence and rationality).

4.3.1 Opponent-independent methods

These methods are based on Q-learning, where policies and state values are computed with game-theoretic solvers for the matrix games arising in the states of the SG [5, 9]. Denoting by $\{Q_{\cdot,k}(x_k,\cdot)\}$ the matrix game arising at time k and given by all the agents' Q-values for state x_k:

$$h_{i,k}(x,\cdot) = \mathbf{solve}_i\{Q_{\cdot,k}(x_k,\cdot)\} \tag{9}$$

$$\begin{aligned} Q_{i,k+1}(x_k,\boldsymbol{u}_k) = Q_{i,k}(x_k,\boldsymbol{u}_k) + \alpha\big[r_{i,k+1} + \\ \gamma\cdot\mathbf{eval}_i\{Q_{\cdot,k}(x_k,\cdot)\} - Q_{i,k}(x_k,\boldsymbol{u}_k)\big] \end{aligned} \tag{10}$$

$\mathbf{solve}_i$ returns the i'th agent's part of some type of equilibrium (a strategy), and $\mathbf{eval}_i$ gives the agent's expected return at this equilibrium. 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 (this problem is similar to the one discussed in Section 4.1).

The learning goal is the convergence to a set of policies that contain the equilibrium strategies in every state. The updates use the Q-tables of all the agents. So, each agent needs to model the Q-tables of the other agents. It can do that by applying (10). This requires two assumptions: that all agents use the same algorithm, and that all actions and rewards are observable.

Particular instances of $\mathbf{solve}$ and $\mathbf{eval}$ for *Nash Q-learning* [8] are:

$$\mathbf{eval}_i\{Q_{\cdot,k}(x_k,\cdot)\} = V_i(x_k,\mathbf{NE}\{Q_{\cdot,k}(x_k,\cdot)\}) \tag{11}$$

$$\mathbf{solve}_i\{Q_{\cdot,k}(x_k,\cdot)\} = \mathbf{NE}_i\{Q_{\cdot,k}(x_k,\cdot)\} \tag{12}$$

where $\mathbf{NE}$ computes a Nash equilibrium, $\mathbf{NE}_i$ is agent i's strategy component of this equilibrium, and $V_i(x_k,\mathbf{NE}\{Q_{\cdot,k}(x_k,\cdot)\})$ is the expected return of agent i from x_k under this equilibrium. *Correlated Q-learning* (CE-Q) [9] and *asymmetric Q-learning* [26] work 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.

Equilibria can also be combined with the Value And Policy Search (VAPS) gradient method [27], leading to *multi-agent VAPS* [28].

4.3.2 Opponent-tracking methods

These algorithms adapt to learned models of nonstationary opponent policies without explicitly considering convergence. Actions of other agents have to be observable. The *Non-Stationary Converging Policies* (NSCP) algorithm computes a best-response to the models and uses it in estimating value functions [29], whereas *Hyper-Q* incorporates the models in the state vector, and learns on their basis [30].

4.3.3 Opponent-aware methods

These methods typically do consider convergence as well as adaptation to other agents. *Win-or-Learn-Fast Policy Hill-Climbing* (WoLF-PHC) combines the basic Q-learning update rule (2) with a gradient-based policy update:

$$\begin{aligned} h_{i,k+1}(x_k,u_i) &= h_{i,k}(x_k,u_i) + \\ &\begin{cases} \delta_{i,k} & \text{if } u_i = \arg\max\limits_{\tilde{u}_i} Q_{i,k+1}(x_k,\tilde{u}_i) \\ -\frac{\delta_{i,k}}{|U_i|-1} & \text{otherwise} \end{cases} \end{aligned} \tag{13}$$

The gradient step $\delta_{i,k}$ is larger when the agent is losing than when it is winning. The win criterion 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 [31]. 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 *Extended Optimal Response* (EXORL) heuristic applies a similar 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 [32].

Environment-Independent Reinforcement Acceleration (EIRA) pushes policies onto, and pops policies from, a policy stack in such a way that long-term reinforcement improvements are guaranteed [33]. EIRA does not make any assumptions on the environment and on the other agents. In this sense, it is very general. However, it may not be able to take advantage of the task's structure.

Remarks

Much research in this area focuses on repeated games. In such a case, one of the essential properties of RL , the delayed reward, is lost. However, the learning problem is still nonstationary due to the dynamic behavior of the agents that play the repeated game. Methods in this category can also be classified into opponent-tracking methods, which aim at adapting to learned models of the opponents' behavior [11] and opponent-aware methods, which target convergence as well [7, 34]. Note that opponent-tracking methods do not necessarily converge to stationary strategies.

Static, repeated games represent a limited set of applications, among which are negotiation, auctions, and bartering. These algorithms provide valuable theoretical results, which, however, do not necessarily carry over to the dynamical SG case.

5 Conclusions and future perspectives

We have reviewed the challenges of multi-agent reinforcement learning and the methods to address them. More general open problems and an outlook are given next.

First, the stage-wise application of game-theoretic techniques may not be the most suitable approach, given that 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 [3, 35]. We expect that tools developed in the area of automatic control will play an important role in the analysis and synthesis of the learning process as a whole (i.e., environment and learning dynamics). In addition, this framework can incorporate prior knowledge on bounds for imperfect observations, such as noise-corrupted variables.

Second, the issue of a suitable learning goal requires additional research. Stability of the learning process is desirable, because the behavior of stable agents is more amenable to analysis and meaningful performance guarantees. Adaptation to the other agents is desirable because their dynamics are generally unpredictable. Therefore, a good multi-agent learning goal must include both components. This means that MARL algorithms should not be purely opponent-independent nor purely opponent-tracking. The control-theoretic concept of robustness can help integrate stability and adaptation into a unified goal. If a learning algorithm is robustly stable with respect to nonstationarity in the other agents, it will converge while allowing for bounded changes in the behavior of these agents.

From a practical viewpoint, a realistic learning goal should include bounds on the transient performance, in addition to the usual asymptotic requirements. Example of such bounds include maximum time constraints for reaching a desired performance level, or a lower bound on instantaneous performance levels. The authors of [10] and [11] already made first steps in this direction.

Third, scalability is an important concern for MARL . Most algorithms require explicit tabular representations of the Q-function and possibly of the policy. Due to this, the computational requirements of the algorithms scale exponentially with the number of state and action variables, and therefore with the number of agents. This also has negative consequences for the learning time and convergence speed. Consider, for instance, extending the grid world of Example 3) to 10×10 cells and the number of agents from two to three. The size of the Q-table that each agent needs to store then becomes $(10 \cdot 10)^3 \cdot 5^3 = 125$ million entries. A similar problem arises when state or action variables are continuous. In this case, tabular storage of Q-values is impossible.

In our view, significant progress in the field of multi-agent learning can be achieved by a more intensive cross-fertilization between the fields of machine learning, game theory and control theory.

Acknowledgement
This research is financially supported by Senter, Ministry of Economic Affairs of the Netherlands, within the BSIK project "Interactive Collaborative Information Systems" (grant no. BSIK03024).

References

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

[2] Y. Shoham, R. Powers, and T. Grenager, "Multi-Agent Reinforcement Learning: A Critical Survey", Technical report, Computer Science Dept., Stanford University, California, US, 16 May 2003.

[3] K. Tuyls and A. Nowé, "Evolutionary Game Theory and Multi-Agent Reinforcement Learning", *The Knowledge Engineering Review*, vol. 20, no. 1, pp. 63–90, 2005.

[4] G. Chalkiadakis, "Multiagent Reinforcement Learning: Stochastic Games with Multiple Learning Players", Technical report, Dept. of Computer Science, University of Toronto, Canada, 25 March 2003.

[5] M. Bowling, *Multiagent Learning in the Presence of Agents with Limitations*, PhD thesis, Computer Science Dept., Carnegie Mellon University, Pittsburgh, US, May 2003.

[6] C. J. C. H. Watkins and P. Dayan, "Technical Note: Q-Learning", *Machine Learning*, vol. 8, pp. 279–292, 1992.

[7] M. Bowling and M. Veloso, "Multiagent Learning Using a Variable Learning Rate", *Artificial Intelligence*, vol. 136, no. 2, pp. 215–250, 2002.

[8] J. Hu and M. P. Wellman, "Nash Q-Learning for General-Sum Stochastic Games", *Journal of Machine Learning Research*, vol. 4, pp. 1039–1069, 2003.

[9] A. Greenwald and K. Hall, "Correlated-Q Learning", in *Proc. Twentieth International Conference on Machine Learning (ICML-03)*, 21–24 August 2003, pp. 242–249, Washington, US.

[10] M. Bowling, "Convergence and No-Regret in Multiagent Learning", in *Advances in Neural Information Processing*

Systems 17 (NIPS-04), 13–18 December 2004, pp. 209–216, Vancouver, Canada.

[11] R. Powers and Y. Shoham, "New Criteria and a New Algorithm for Learning in Multi-Agent Systems", in *Advances in Neural Information Processing Systems 17 (NIPS-04)*, 2004, pp. 1089–1096, Vancouver, Canada.

[12] 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. Twentieth International Conference on Machine Learning (ICML-03)*, 21–24 August 2003, pp. 83–90, Washington, US.

[13] M. L. Littman, "Value-function Reinforcement Learning in Markov Games", *Journal of Cognitive Systems Research*, vol. 2, pp. 55–66, 2001.

[14] M. Lauer and M. Riedmiller, "An Algorithm for Distributed Reinforcement Learning in Cooperative Multi-Agent Systems", in *Proc. Seventeenth International Conference on Machine Learning (ICML-00)*, 29 June – 2 July 2000, pp. 535–542, Stanford University, US.

[15] C. Boutilier, "Planning, Learning and Coordination in Multiagent Decision Processes", in *Proc. Sixth Conference on Theoretical Aspects of Rationality and Knowledge (TARK-96)*, 17–20 March 1996, pp. 195–210, De Zeeuwse Stromen, The Netherlands.

[16] 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 *Workshop on Cooperative Robotics, 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-02)*, 1 October 2002, pp. 66–73, Lausanne, Switzerland.

[17] C. Guestrin, M. G. Lagoudakis, and R. Parr, "Coordinated Reinforcement Learning", in *Proc. Nineteenth International Conference on Machine Learning (ICML-02)*, 8–12 July 2002, pp. 227–234, Sydney, Australia.

[18] N. Vlassis, "A Concise Introduction to Multiagent Systems and Distributed AI", Technical report, University of Amsterdam, The Netherlands, September 2003, URL: http://www.science.uva.nl/ vlassis/cimasdai/cimasdai.pdf.

[19] F. Fischer, M. Rovatsos, and G. Weiss, "Hierarchical Reinforcement Learning in Communication-Mediated Multiagent Coordination", in *Proc. 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-04)*, 19–23 August 2004, pp. 1334–1335, New York, US.

[20] C. Claus and C. Boutilier, "The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems", in *Proc. 15th National Conference on Artificial Intelligence and 10th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI-98)*, 26–30 July 1998, pp. 746–752, Madison, US.

[21] S. Kapetanakis and D. Kudenko, "Reinforcement Learning of Coordination in Cooperative Multi-Agent Systems", in *Proc. 18th National Conference on Artificial Intelligence and 14th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI-02)*, 28 July – 1 August 2002, pp. 326–331, Menlo Park, US.

[22] X. Wang and T. Sandholm, "Reinforcement Learning to Play an Optimal Nash Equilibrium in Team Markov Games", in *Advances in Neural Information Processing Systems 15 (NIPS-02)*, 9–14 December 2002, pp. 1571–1578, Vancouver, Canada.

[23] S. Sen, M. Sekaran, and J. Hale, "Learning to Coordinate without Sharing Information", in *Proc. 12th National Conference on Artificial Intelligence (AAAI-94)*, 31 July – 4 August 1994, pp. 426–431, Seattle, US.

[24] M. J. Matarić, "Learning in Multi-Robot Systems", in G. Weiß and S. Sen, editors, *Adaptation and Learning in Multi–Agent Systems*, pp. 152–163. Springer Verlag, 1996.

[25] R. H. Crites and A. G. Barto, "Improving Elevator Performance Using Reinforcement Learning", in *Advances in Neural Information Processing Systems*, volume 8, 1996, pp. 1017–1023.

[26] V. Könönen, "Asymmetric Multiagent Reinforcement Learning", in *Proc. IEEE/WIC International Conference on Intelligent Agent Technology (IAT-03)*, 13–17 October 2003, pp. 336–342, Halifax, Canada.

[27] L. Baird and A. Moore, "Gradient Descent for General Reinforcement Learning", in *Advances in Neural Information Processing Systems 11 (NIPS-98)*, 30 November – 5 December 1998, pp. 968–974, Denver, US.

[28] V. Könönen, "Gradient Based Method for Symmetric and Asymmetric Multiagent Reinforcement Learning", in *Proc. 4th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL-03)*, 21–23 March 2003, pp. 68–75, Hong Kong, China.

[29] M. Weinberg and J. S. Rosenschein, "Best-Response Multiagent Learning in Non-Stationary Environments", in *Proc. 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-04)*, 19-23 August 2004, pp. 506–513, New York, US.

[30] G. Tesauro, "Extending Q-Learning to General Adaptive Multi-Agent Systems", in *Advances in Neural Information Processing Systems 16 (NIPS-03)*, 8–13 December 2003, Vancouver and Whistler, Canada.

[31] B. Banerjee and J. Peng, "Adaptive Policy Gradient in Multiagent Learning", in *Proc. 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03)*, 14–18 July 2003, pp. 686–692, Melbourne, Australia.

[32] N. Suematsu and A. Hayashi, "A multiagent reinforcement learning algorithm using extended optimal response", in *Proc. 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-02)*, 15–19 July 2002, pp. 370–377, Bologna, Italy.

[33] J. Schmidhuber, "A General Method for Multi-Agent Reinforcement Learning in Unrestricted Environments", in *Working Notes AAAI Symposium on Adaptation, Coevolution and Learning in Multiagent Systems*, 25–27 March 1996, pp. 84–87, Stanford University, US.

[34] S. Singh, M. Kearns, and Y. Mansour, "Nash Convergence of Gradient Dynamics in General-Sum Games", in *Proc. 16th Conference on Uncertainty in Artificial Intelligence (UAI-00)*, 30 June – 3 July 2000, pp. 541–548, San Francisco, US.

[35] J. M. Vidal, "Learning in Multiagent Systems: An Introduction from a Game-Theoretic Perspective", in *Adaptive Agents: Lecture Notes in Artificial Intelligence*, volume 2636, pp. 202–215. Springer Verlag, August 2003.