Chapter 14
Game Theory and Multi-agent Reinforcement Learning

Ann Nowé, Peter Vrancx, and Yann-Michaël De Hauwere

Abstract. Reinforcement Learning was originally developed for Markov Decision Processes (MDPs). It allows a single agent to learn a policy that maximizes a possibly delayed reward signal in a stochastic stationary environment. It guarantees convergence to the optimal policy, provided that the agent can sufficiently experiment and the environment in which it is operating is Markovian. However, when multiple agents apply reinforcement learning in a shared environment, this might be beyond the MDP model. In such systems, the optimal policy of an agent depends not only on the environment, but on the policies of the other agents as well. These situations arise naturally in a variety of domains, such as: robotics, telecommunications, economics, distributed control, auctions, traffic light control, etc. In these domains multi-agent learning is used, either because of the complexity of the domain or because control is inherently decentralized. In such systems it is important that agents are capable of discovering good solutions to the problem at hand either by coordinating with other learners or by competing with them. This chapter focuses on the application reinforcement learning techniques in multi-agent systems. We describe a basic learning framework based on the economic research into game theory, and illustrate the additional complexity that arises in such systems. We also described a representative selection of algorithms for the different areas of multi-agent reinforcement learning research.

14.1 Introduction

The reinforcement learning techniques studied throughout this book enable a single agent to learn optimal behavior through trial-and-error interactions with its environment. Various RL techniques have been developed which allow an agent to optimize

Ann Nowé · Peter Vrancx · Yann-Michaël De Hauwere
Vrije Universiteit Brussel
e-mail: {anowe,pvrancx,ydehauwe}@vub.ac.be

M. Wiering and M. van Otterlo (Eds.): Reinforcement Learning, ALO 12, pp. 441–470.
springerlink.com

its behavior in a wide range of circumstances. However, when multiple learners simultaneously apply reinforcement learning in a shared environment, the traditional approaches often fail.

In the multi-agent setting, the assumptions that are needed to guarantee convergence are often violated. Even in the most basic case where agents share a stationary environment and need to learn a strategy for a single state, many new complexities arise. When agent objectives are aligned and all agents try to maximize the same reward signal, coordination is still required to reach the global optimum. When agents have opposing goals, a clear optimal solution may no longer exist. In this case, an equilibrium between agent strategies is usually searched for. In such an equilibrium, no agent can improve its payoff when the other agents keep their actions fixed.

When, in addition to multiple agents, we assume a dynamic environment which requires multiple sequential decisions, the problem becomes even more complex. Now agents do not only have to coordinate, they also have to take into account the current state of their environment. This problem is further complicated by the fact that agents typically have only limited information about the system. In general, they may not be able to observe actions or rewards of other agents, even though these actions have a direct impact on their own rewards and their environment. In the most challenging case, an agent may not even be aware of the presence of other agents, making the environment seem non-stationary. In other cases, the agents have access to all this information, but learning in a fully joint state-action space is in general impractical, both due to the computational complexity and in terms of the coordination required between the agents. In order to develop a successful multi-agent approach, all these issues need to be addressed. Figure 14.1 depicts a standard model of Multi-Agent Reinforcement Learning.

Despite the added learning complexity, a real need for multi-agent systems exists. Often systems are inherently decentralized, and a central, single agent learning approach is not feasible. This situation may arise because data or control is physically distributed, because multiple, possibly conflicting, objectives should be met, or simply because a single centralized controller requires to many resources. Examples of such systems are multi-robot set-ups, decentralized network routing, distributed load-balancing, electronic auctions, traffic control and many others.

The need for adaptive multi-agent systems, combined with the complexities of dealing with interacting learners has led to the development of a multi-agent reinforcement learning field, which is built on two basic pillars: the reinforcement learning research performed within AI, and the interdisciplinary work on game theory. While early game theory focused on purely competitive games, it has since developed into a general framework for analyzing strategic interactions. It has attracted interest from fields as diverse as psychology, economics and biology. With the advent of multi-agent systems, it has also gained importance within the AI community and computer science in general. In this chapter we discuss how game theory provides both a means to describe the problem setting for multi-agent learning and the tools to analyze the outcome of learning.

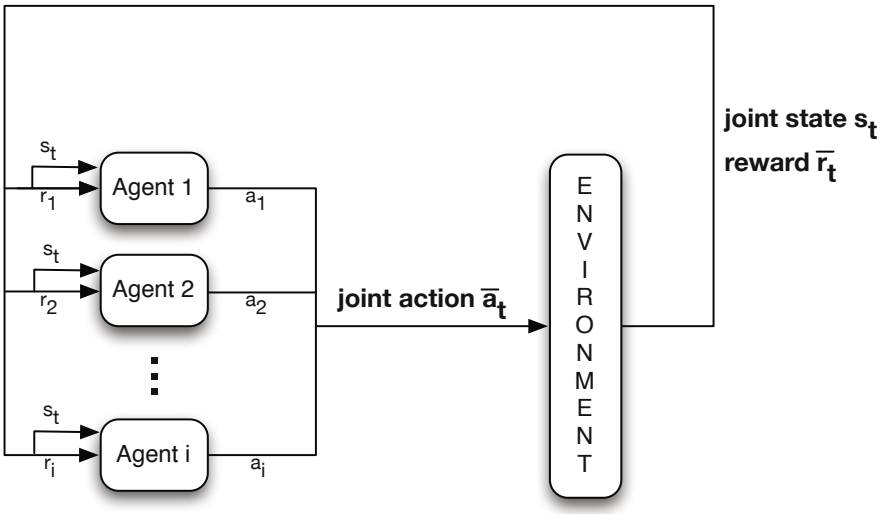

Fig. 14.1 Multiple agents acting in the same environment

The multi-agent systems considered in this chapter are characterized by strategic interactions between the agents. By this we mean that the agents are autonomous entities, who have individual goals and independent decision making capabilities, but who also are influenced by each other's decisions. We distinguish this setting from the approaches that can be regarded as distributed or parallel reinforcement learning. In such systems multiple learners collaboratively learn a single objective. This includes systems were multiple agents update the policy in parallel (Mariano and Morales, 2001), swarm based techniques (Dorigo and Stützle, 2004) and approaches dividing the learning state space among agents (Steenhaut et al, 1997). Many of these systems can be treated as advanced exploration techniques for standard reinforcement learning and are still covered by the single agent theoretical frameworks, such as the framework described in (Tsitsiklis, 1994). The convergence of the algorithms remain valid as long as outdated information is eventually discarded. For example, it allows to use outdated Q-values in the max-operator in the right hand side of standard Q-learning update rule (described in Chapter 1). This is particularly interesting when he Q-values are belonging to to different agents each exploring their own part of the environment and only now and then exchange their Q-values. The systems covered by this chapter, however, go beyond the standard single agent theory, and as such require a different framework.

An overview of multi-agent research based on strategic interactions between agents is given in Table 14.1. The techniques listed are categorized based on their

applicability and kind of information they use while learning in a multi-agent system. We distinguish between techniques for stateless games, which focus on dealing with multi-agent interactions while assuming that the environment is stationary, and Markov game techniques, which deal with both multi-agent interactions and a dynamic environment. Furthermore, we also show the information used by the agents for learning. Independent learners learn based only on their own reward observation, while joint action learners also use observations of actions and possibly rewards of the other agents.

Table 14.1 Overview of current MARL approaches. Algorithms are classified by their applicability (common interest or general Markov games) and their information requirement (scalar feedback or joint-action information).

		Game setting		
		Stateless Games	Team Markov Games	General Markov Games
Information Requirement	Independent Learners	Stateless Q-learning Learning Automata IGA FMQ Commitment Sequences Lenient Q-learners	Policy Search Policy Gradient	MG-ILA (WoLF-)PG Learning of Coordination Independent RL CQ-learning
	Joint Action Learners		Distributed- Q Sparse Tabular Q Utile Coordination	Nash-Q Friend-or-Foe Q Asymmetric Q Joint Action Learning Correlated-Q

In the following section we will describe the repeated games framework. This setting introduces many of the complexities that arise from interactions between learning agents. However, the repeated game setting only considers static, stateless environments, where the learning challenges stem only from the interactions with other agents. In Section 14.3 we introduce Markov Games. This framework generalizes the Markov Decision Process (MDP) setting usually employed for single agent RL. It considers both interactions between agents and a dynamic environment. We explain both value iteration and policy iteration approaches for solving these Markov games. Section 14.4 describes the current state of the art in multi-agent research, which takes the middle ground between independent learning techniques and Markov game techniques operating in the full joint-state joint-action space. Finally in Section 14.5, we shortly describe other interesting background material.

14.2 Repeated Games

14.2.1 Game Theory

The central idea of game theory is to model strategic interactions as a game between a set of players. A game is a mathematical object, which describes the consequences of interactions between player strategies in terms of individual payoffs. Different representations for a game are possible. For example,traditional AI research often focusses on the *extensive form* games, which were used as a representation of situations where players take turns to perform an action. This representation is used, for instance, with the classical minimax algorithm (Russell and Norvig, 2003). In this chapter, however, we will focus on the so called *normal form* games, in which game players simultaneously select an individual action to perform. This setting is often used as a testbed for multi-agent learning approaches. Below we the review basic game theoretic terminology and define some common solution concepts in games.

14.2.1.1 Normal Form Games

Definition 14.1. A normal form game is a tuple $(n, A_{1,\ldots,n}, R_{1,\ldots,n})$, where

- $1,\ldots,n$ is a collection of participants in the game, called players;
- A_k is the individual (finite) set of actions available to player k;
- $R_k : A_1 \times \ldots \times A_n \to \mathbb{R}$ is the individual reward function of player k, specifying the expected payoff he receives for a play $\mathbf{a} \in A_1 \times \ldots \times A_n$.

A game is played by allowing each player k to independently select an individual action a from its private action set A_k.The combination of actions of all players constitute a *joint action* or *action profile* $\mathbf{a}$ from the joint action set $\mathbb{A} = A_1 \times \ldots \times A_n$. For each joint action $\mathbf{a} \in \mathbb{A}$, $R_k(\mathbf{a})$ denotes agent k's expected payoff.

Normal form games are represented by their *payoff matrix*. Some typical 2-player games are given in Table 14.2. In this case the action selected by player 1 refers to a row in the matrix, while that of player 2 determines the column. The corresponding entry in the matrix then gives the payoffs player 1 and player 2 receive for the play. Players 1 and 2 are also referred to as the row and the column player, respectively. Using more dimensional matrices n-player games can be represented where each entry in the matrix contains the payoff for each of the agents for the corresponding combination of actions.

A *strategy* $\sigma_k : A_k \to [0,1]$ is an element of $\mu(A_k)$, the set of probability distributions over the action set A_k of player k . A strategy is called pure if $\sigma_k(a) = 1$ for some action $a \in A_k$ and 0 for all other actions, otherwise it is called a *mixed strategy*. A *strategy profile* $\sigma = (\sigma_1, \ldots, \sigma_n)$ is a vector of strategies, containing one strategy for each player. If all strategies in σ are pure, the strategy profile corresponds to a joint action $\mathbf{a} \in \mathbb{A}$. An important assumption which is made in normal form games is

that the expected payoffs are linear in the player strategies, i.e. the expected reward for player k for a strategy profile σ is given by:

$$R_k(\sigma) = \sum_{\mathbf{a} \in \mathbb{A}} \prod_{j=1}^{n} \sigma_j(a_j) R_k(\mathbf{a})$$

with a_j the action for player j in the action profile $\mathbf{a}$.

14.2.1.2 Types of Games

Depending on the reward functions of the players, a classifications of games can be made. When all players share the same reward function, the game is called a *identical payoff* or *common interest* game. If the total of all players rewards adds up to 0 the game is called a *zero-sum game*. In the latter games wins for certain players translate to losses for other players with opposing goals. Therefore these games are also referred to as pure competitive games. When considering games without special restrictions we speak of *general sum games*.

Table 14.2 Examples of 2-player, 2-action games. From left to right: (a) Matching pennies, a purely competitive (zero-sum) game. (b) The prisoner's dilemma, a general sum game. (c) The coordination game, a common interest (identical payoff) game. (d) Battle of the sexes, a coordination game where agents have different preferences) Pure Nash equilibria are indicated in bold.

	a1	a2
a1	(1,-1)	(-1,1)
a2	(-1,1)	(1,-1)

(a)

	a1	a2
a1	(5,5)	(0,10)
a2	(10,0)	**(1,1)**

(b)

	a1	a2
a1	**(5,5)**	(0,0)
a2	(0,0)	**(10,10)**

(c)

	a1	a2
a1	**(2,1)**	(0,0)
a2	(0,0)	**(1,2)**

(d)

Examples of these game types can be seen in Table 14.2. The first game in this table, named matching pennies, is an example of a strictly competitive game. This game describes a situation where the two players must each, individually, select one side of a coin to show (i.e. Heads or Tails). When both players show the same side, player one wins and is paid 1 unit by player 2. When the coins do not match, player 2 wins and receives 1 unit from player 1. Since both players are betting against each other, one player's win automatically translates in the other player's loss, therefore this is a zero-sum game.

The second game in Table 14.2, called the prisoner's dilemma, is a general sum game. In this game, 2 criminals have been apprehended by the police for committing a crime. They both have 2 possible actions: cooperate with each other and deny the crime (action a1), or defect and betray the other, implicating him in the crime (action a2). If both cooperate and deny the crime, the police have insufficient evidence and they get a minimal sentence, which translates to a payoff of 5 for both. If one player cooperates, but the other one defects, the cooperator takes all the blame

(payoff 0), while the defector escapes punishment (payoff 10). Should both players defect, however, they both receive a large sentence (payoff 1). The issue in this game is that the cooperate action is *strictly dominated* by the defect action: no matter what action the other player chooses, to defect always gives the highest payoff. This automatically leads to the *(defect, defect)* outcome, despite the fact that both players could simultaneously do better by both playing cooperate.

The third game in Table 14.2 is a common interest game. In this case both players receive the same payoff for each joint action. The challenge in this game is for the players to coordinate on the optimal joint action. Selecting the wrong joint action gives a suboptimal payoff and failing to coordinate results in a 0 payoff.

The fourth game, Battle of the sexes, is another example of a coordination game. Here however, the players get individual rewards and prefer different outcomes. Agent 1 prefers (a1,a1), whereas agent 2 prefers (a2,a2). In addition to the coordination problem, the players now also have to agree on which of the preferred outcomes.

Of course games are not restricted to only two actions but can have any number of actions. In Table 14.3 we show some 3-action common interest games. In the first, the climbing game from (Claus and Boutilier, 1998), the Nash equilibria are surrounded by heavy penalties. In the second game, the penalties are left as a parameter $k < 0$. The smaller k, the more difficult it becomes to agree through learning on the preferred solution ((a1,a1) and (a3,a3)) (The dynamics of these games using a value-iteration approach are analyzed in (Claus and Boutilier, 1998), see also Section 14.2.2).

Table 14.3 Examples of 2-player, 3-action games. From left to right: (a) Climbing game (b) The penalty game, where $k \leq 0$. Both games are of the common interest type. Pure Nash equilibria are indicated in bold.

	a1	a2	a3
a1	**(11,11)**	(-30,-30)	(0,0)
a2	(-30,-30)	**(7,7)**	(6,6)
a3	(0,0)	(0,0)	(5,5)

(a)

	a1	a2	a3
a1	**(10,10)**	(0,0)	*(k,k)*
a2	(0,0)	**(2,2)**	(0,0)
a3	*(k,k)*	(0,0)	**(10,10)**

(b)

14.2.1.3 Solution Concepts in Games

Since players in a game have individual reward functions which are dependent on the actions of other players, defining the desired outcome of a game is often not clearcut. One cannot simply expect participants to maximize their payoffs, as it may not be possible for all players to achieve this goal at the same time. See for example the Battle of the sexes game in Table 14.2(d).

An important concept for such learning situations, is that of a *best response*. When playing a best response, a player maximizes his payoff with respect to the current strategies of his opponents in the game. That is, it is not possible for the player to improve his reward if the other participants in the game keep their strategies fixed. Formally, we can define this concept as follows:

Definition 14.2. Let $\sigma = (\sigma_1, \ldots, \sigma_n)$ be a strategy profile and let σ_{-k} denote the same strategy profile but without the strategy σ_k of player k. A strategy $\sigma_k^* \in \mu(A_k)$ is then called a best response for player k, if following holds:

$$R_k(\sigma_{-k} \cup \sigma_k^*) \geq R_k(\sigma_{-k} \cup \sigma_k') \;\; \forall \sigma_k' \in \mu(A_k)$$

where $\sigma_{-k} \cup \sigma_k'$ denotes the strategy profile where all agents play the same strategy as they play in σ except agent k who plays σ_k', i.e. $(\sigma_1, \ldots, \sigma_{k-1}, \sigma_k', \sigma_{k+1}, \ldots, \sigma_n)$.

A central solution concept in games, is the *Nash equilibrium* (NE). In a Nash equilibrium, the players all play mutual best replies, meaning that each player uses a best response to the current strategies of the other players. Nash (Nash, 1950) proved that every normal form game has at least 1 Nash equilibrium, possibly in mixed strategies. Based on the concept of best response we can define a Nash equilibrium as:

Definition 14.3. A strategy profile $\sigma = (\sigma_1, \ldots, \sigma_n)$ is called a Nash equilibrium if for each player k, the strategy σ_k is a best response to the strategies of the other players σ_{-k}.

Thus, when playing a Nash equilibrium, no player in the game can improve his payoff by unilaterally deviating from the equilibrium strategy profile. As such no player has an incentive to change his strategy, and multiple players have to change their strategy simultaneously in order to escape the Nash equilibrium.

In common interest games such as the coordination in Table 14.2(c), the Nash equilibrium corresponds to a local optimum for all players, but it does not necessarily correspond to the global optimum. This can clearly be seen in the coordination game, where we have 2 Nash equilibria: the play $(a1,a1)$ which gives both players a reward of 5 and the global optimum $(a2,a2)$ which results in a payoff of 10.

The prisoner's dilemma game in Table 14.2 shows that a Nash equilibrium does not necessarily correspond to the most desirable outcome for all agents. In the unique Nash equilibrium both players prefer the 'defect' action, despite the fact that both would receive when both are cooperating. The cooperative outcome is not a Nash equilibrium, however, as in this case both players can improve their payoff by switching to the 'defect' action.

The first game, matching pennies, does not have a pure strategy Nash equilibrium, as no pure strategy is a best response to another pure best response. Instead the Nash equilibrium for this game is for both players to choose both sides with equal probability. That is, the Nash strategy profile is ((1/2,1/2),(1/2,1/2)).

14.2.2 Reinforcement Learning in Repeated Games

The games described above are often used as test cases for multi-agent reinforcement learning techniques. Unlike in the game theoretical setting, agents are not assumed to have full access to the payoff matrix. In the reinforcement learning setting, agents are taken to be players in a normal form game, which is played repeatedly, in order to improve their strategy over time.

It should be noted that these *repeated games* do not yet capture the full multi-agent reinforcement learning problem. In a repeated game all changes in the expected reward are due to changes in strategy by the players. There is no changing environment state or state transition function external to the agents. Therefore, repeated games are sometimes also referred to as *stateless games*. Despite this limitation, we will see further in this section that these games can already provide a challenging problem for independent learning agents, and are well suited to test coordination approaches. In the next section, we will address the Markov game framework which does include a dynamic environment.

A number of different considerations have to be made when dealing with reinforcement learning in games. As is common in RL research, but contrary to traditional economic game theory literature, we assume that the game being played is initially unknown to the agents, i.e. agents do not have access to the reward function and do not know the expected reward that will result from playing a certain (joint) action. However, RL techniques can still differ with respect to the observations the agents make. Moreover, we also assume that the game payoffs can be stochastic, meaning that a joint action does not always result in the same deterministic reward for each agent. Therefore, actions have to be sampled repeatedly.

Since expected rewards depend on the strategy of all agents, many multi-agent RL approaches assume that the learner can observe the actions and/or rewards of all participants in the game. This allows the agent to model its opponents and to explicitly learn estimates over joint actions. It could be argued however, that this assumption is unrealistic, as in multi-agent systems which are physically distributed this information may not be readily available. In this case the RL techniques must be able to deal with the non-stationary rewards caused by the influence of the other agents. As such, when developing a multi-agent reinforcement learning application it is important to consider the information available in a particular setting in order to match this setting with an appropriate technique.

14.2.2.1 Goals of Learning

Since it is in general impossible for all players in a game to maximize their payoff simultaneously, most RL methods attempt to achieve Nash equilibrium play. However, a number of criticisms can be made of the Nash equilibrium as a solution concept for learning methods. The first issue is that Nash equilibria need not be unique, which leads to an equilibrium selection problem. In general, multiple Nash equilibria can exist for a single game. These equilibria can also differ in the

payoff they give to the players. This means that a method learning Nash equilibria, cannot guarantee a unique outcome or even a unique payoff for the players. This can be seen in the coordination game of Table 14.2(c), where 2 Nash equilibria exist, one giving payoff 5 to the agents, and the other giving payoff 10. The game in Table 14.3(b) also has multiple NE, with $(a1,a1)$ and $(a3,a3)$ being the 2 optimal ones. This results in a coordination problem for learning agents, as both these NE have the same quality.

Furthermore, since the players can have different expected payoffs even in an equilibrium play, the different players may also prefer different equilibrium outcomes, which means that care should be taken to make sure the players coordinate on a single equilibrium. This situation can be observed in the Battle of the sexes game in Table 14.2(d), where 2 pure Nash equilibria exist, but each player prefers a different equilibrium outcome.

Another criticism is that a Nash Equilibrium does not guarantee optimality. While playing a Nash equilibrium assures that no single player can improve his payoff by unilaterally changing its strategy, it does not guarantee that the players globally maximize their payoffs, or even that no play exists in which the players simultaneously do better. It is possible for a game to have non-Nash outcomes, which nonetheless result in a higher payoff to all agents than they would receive for playing a Nash equilibrium. This can be seen for example in the prisoner's dilemma in Table 14.2(c).

While often used as the main goal of learning, Nash equilibria are not the only possible solution concept in game theory. In part due to the criticisms mentioned above, a number of alternative solution concepts for games have been developed. These alternatives include a range of other equilibrium concepts, such as the *Correlated Equilibrium*(CE))(Aumann, 1974), which generalizes the Nash equilibrium concept, or the *Evolutionary Stable Strategy* (ESS)(Smith, 1982), which refines the Nash equilibrium. Each of these equilibrium outcomes has its own applications and (dis)advantages. Which solution concept to use depends on the problem at hand, and the objective of the learning algorithm. A complete discussion of possible equilibrium concepts is beyond the scope of this chapter. We focus on the Nash equilibrium and briefly mention regret minimization as these are the approaches most frequently observed in the multi-agent learning literature. A more complete discussion of solution concepts can be found in many textbooks, e.g. (Leyton-Brown and Shoham, 2008).

Before continuing, we mention one more evaluation criterion, which is regularly used in repeated games: the notion of *regret*. Regret is the difference between the payoff an agent realized and the maximum payoff the agent could have obtained using some fixed strategy. Often the fixed strategies that one compares the agent performance to, are simply the pure strategies of the agent. In this case, the total regret of the agent is the accumulated difference between the obtained reward and the reward the agent would have received for playing some fixed action. For an agent k, given the history of play at time T, this is defined as:

$$\mathscr{R}_T = max_{a \in A_k} \sum_{t=1}^{T} R_k(\mathbf{a}_{-k}(t) \cup \{a\}) - R_k(\mathbf{a}(t)), \tag{14.1}$$

where $\mathbf{a}(t)$ denotes the joint action played at time t and $\mathbf{a}_{-k}(t) \cup \{a\}$ denotes the same joint action but with player k playing action a. Most regret based learning approaches attempt to minimize the average regret $\mathscr{R}_T/T$ of the learner. Exact calculation of this regret requires knowledge of the reward function and observation of the actions of other agents in order to determine the $R_k(\mathbf{a}_{-k}(t) \cup \{a\})$ term. If this information is not available, regret has to be estimated from previous observations. Under some assumptions regret based learning can been shown to converge to some form of equilibrium play (Foster and Young, 2003; Hart and Mas-Colell, 2001).

14.2.2.2 Q-Learning in Games

A natural question to ask is what happens when agents use a standard, single-agent RL technique to interact in a game environment. Early research into multi-agent RL focussed largely on the application of Q-learning to repeated games. In this so called independent or uninformed setting, each player k keeps an individual vector of estimated Q-values $Q_k(a)$, $a \in A_k$. The players learn Q-values over their own action set and do not use any information on other players in the game. Since there is no concept of environment state in repeated games, a single vector of estimates is sufficient, rather than a full table of state-action pairs, and the standard Q-learning update is typically simplified to its stateless version:

$$Q(a) \leftarrow Q(a) + \alpha[r(t) - Q(a)] \tag{14.2}$$

In (Claus and Boutilier, 1998) the dynamics of stateless Q-learning in repeated normal form common interest games are empirically studied.The key questions here are: is simple Q-learning still guaranteed to converge in a multi-agent setting, and if so, does it converge to (the optimal) equilibrium. It also relates independent Q-learners to joint action learners (see below) and investigates how the rates of convergence and limit points are influenced by the game structures and action selection strategies. In a related branch of research (Tuyls and Nowé, 2005; Wunder et al, 2010) the dynamics of independent Q-learning are studied using techniques from *evolutionary game theory* (Smith, 1982).

While independent Q-learners were shown to reach equilibrium play under some circumstances, they also demonstrated a failure to coordinate in some games, and even failed to converge altogether in others.

They compared joint action learners to independent learners. In the former the agents learn Q-values for all joint actions, with other words each agent j leans a Q-value for all a in A. The action selection is done by each agent individually based on the believe the agents has about the other agents strategy. Equation 14.3 expresses that the Q-value of the joint action is weighted according to the probability the other agents will select some particular value. The Expected Values (EV) can then be used in combination with any action selection technique. Claus and Boutilier showed

experimentally using the games of table 2, that joint action learners and independent learners using a Boltzmann exploration strategy with decreasing temperature behave very similar. These learners have been studied from an evolutionary game theory point of view in (Tuyls and Nowé, 2005) and is has been shown that these learners will converge to evolutionary stable NE which are not necessarily pareto optimal.

$$EV(a^i) = \sum_{a^{-i} \in A_{-i}} Q(a^{-i} \cup \{a^i\}) \prod_{j \neq i} \{Pr^i_{a^{-i}[j]}\} \tag{14.3}$$

However the learners have difficulties to reach the optimal NE, and more sophisticated exploration strategies are needed to increase the probability of converging to the optimal NE. The reason that simple exploration strategies are not sufficient is mainly due to the fact that the actions involved in the optimal NE often lead to much lower payoff when combined with other actions, the potential quality of the action is therefore underestimated. For example in game 2a the action a1 of the row player, will only lead to the highest reward 11 when combined with action a1 of the column player. During the learning phase, agents are still exploring and action a1 will also be combined with actions a2 and a3. As a results the agents will often settle for the more "safe" NE (a2,a2). A similar behavior is observed in game 2b, since miscoordination on the 2 NE is punished, the bigger the penalty ($k_i 0$) the more difficult it become for the agents to reach either of the optimal NE. This also explains why independent learners are generally not able to converge to a NE when they are allowed to use any, including a random exploration strategy. Whereas in single agent Q-learning, the particular exploration strategy does not affect the eventual convergence (Tsitsiklis, 1994) this no longer holds in a MAS setting.

The limitations of single-agent Q-learning have lead to a number of extensions of the Q-learning algorithm for use in repeated games. Most of these approaches focus on coordination mechanisms allowing Q-learners to reach the optimal outcome in common interest games. The frequency maximum Q-learning (FMQ) algorithm (Kapetanakis and Kudenko, 2002), for example, keeps a frequency value $freq(R^*,a)$ indicating how often the maximum reward so far (R^*) has been observed for a certain action a. This value is then used as a sort of heuristic which is added to the Q-values. Instead of using Q-values directly, the FMQ algorithm relies on following heuristic evaluation of the actions:

$$EV(a) = Q(a) + w.freq(R^*,a).R^*, \tag{14.4}$$

where w is a weight that controls the importance of the heuristic value $freq(R^*,a)R^*$. The algorithm was empirically shown to be able to drive learners to the optimal joint action in common interest games with deterministic payoffs.

In (Kapetanakis et al, 2003) the idea of commitment sequences has been introduced to allow independent learning in games with stochastic payoffs. A commitment sequence is a list of time slots for which an agent is committed to selecting always the same action. These sequences of time slots is generated according to some protocol the agents are aware off. Using this guarantee that at time slots belonging

to the same sequence the agents are committed to always select the same individual action, the agents are able to distinguish between the two sources of uncertainty: the noise on the reward signal and the influence on the reward by the actions taken by the other agents. This allows the agents to deal with games with stochastic payoffs.

A recent overview of multi-agent Q-learning approaches can be found in (Wunder et al, 2010).

14.2.2.3 Gradient Ascent Approaches

As an alternative to the well known Q-learning algorithm, we now list some approaches based on gradient following updates. We will focus on players that employ learning automata (LA) reinforcement schemes. Learning automata are relatively simple policy iterators, that keep a vector action probabilities **p** over the action set A. As is common in RL, these probabilities are updated based on a feedback received from the environment. While initial studies focussed mainly on a single automaton in n-armed bandit settings, RL algorithms using multiple automata were developed to learn policies in MDPs (Wheeler Jr and Narendra, 1986). The most commonly used LA update scheme is called Linear Reward-Penalty and updates the action probabilities as follows:

$$p_i(t+1) = p_i(t) + \lambda_1 b(t)(1-p_i(t)) - \lambda_2(1-r(t))p_i(t) \quad \text{if } a(t) = a_i, \tag{14.5}$$

$$p_j(t+1) = p_j(t) - \lambda_1 r(t)p_j(t) + \lambda_2(1-r(t))(\frac{1}{K-1} - p_j(t)) \quad \text{if } a_j \neq a_i, \tag{14.6}$$

$r(t)$ being the feedback received at time t and K the number of actions in available to the auomaton. λ_1 and λ_2 are constants, called the reward and penalty parameter respectively. Depending on the values of these parameters 3 distinct variations of the algorithm can be considered. When $\lambda_1 = \lambda_2$, the algorithm is referred to as *Linear Reward-Penalty* (L_{R-P}) while it is called *Linear Reward-εPenalty* ($L_{R-\varepsilon P}$) when $\lambda_1 >> \lambda_2$. If $\lambda_2 = 0$ the algorithm is called *Linear Reward-Inaction* (L_{R-I}). In this case, λ_1 is also sometimes called the learning rate:

$$p_i(t+1) = p_i(t) + \lambda_1 r(t)(1-p_i(t)) \quad \text{if } a(t) = a_i \tag{14.7}$$

$$p_j(t+1) = p_j(t) - \lambda_1 r(t)p_j(t) \quad \text{if } a_j \neq a_i \tag{14.8}$$

This algorithm has also been shown to be a special case of the REINFORCE (Williams, 1992) update rules. Despite the fact that all these update rules are derived

from the same general scheme, they exhibit very different learning behaviors. Interestingly, these learning schemes perform well in game contexts, even though they do not require any information (actions, rewards, strategies) on the other players in the game. Each agent independently applies a LA update rule to change the probabilities over its actions. Below we list some interesting properties of LA in game settings. In two-person zero-sum games, the L_{R-I} scheme converges to the Nash equilibrium when this exists in pure strategies, while the $L_{R-\varepsilon P}$ scheme is able to approximate mixed equilibria. In n-player common interest games reward-inaction also converges to a pure Nash equilibrium. In (Sastry et al, 1994), the dynamics of reward-inaction in general sum games are studied. The authors proceed by approximating the update in the automata game by a system of ordinary differential equations. Following properties are found to hold for the L_{R-I} dynamics:

- All Nash equilibria are stationary points.
- All strict Nash equilibria are asymptotically stable.
- All stationary points that are not Nash equilibria are unstable.

Furthermore, in (Verbeeck, 2004) it is shown that an automata team using the reward-inaction scheme will convergence to a pure joint strategy with probability 1 in common as well as conflicting interest games with stochastic payoffs. These results together imply local convergence towards pure Nash equilibria in n-player general-sum games(Verbeeck, 2004). Since NE with higher payoffs are stronger attractors for the LA, the agents are more likely to reach the better NE. Equipped with an exploration strategy with only requires very limited communications, the agents are able to explore the interesting NE without the need for exhaustive exploration and once these are found, different solution concepts can be considered, for example fair solutions alternating between different Pareto optimal solutions.

In (Verbeeck et al, 2005) it has also been shown that these LA based approach is able to converge in a setting where agents take actions asynchronously and the rewards are delayed as is common in load balancing settings or congestion games.

Another gradient technique frequently studied in games is the *Infinitesimal Gradient Ascent* (IGA) family of algorithms (Singh et al, 2000; Bowling and Veloso, 2001; Zinkevich, 2003; Bowling, 2005). In addition to demonstrating Nash equilibrium convergence in a number of repeated game settings, several of these papers also evaluate the algorithms with respect to their regret.

14.3 Sequential Games

While highlighting some of the important issues introduced by learning in a multi-agent environment, the traditional game theory framework does not capture the full complexity of multi-agent reinforcement learning. An important part of the reinforcement learning problem is that of making sequential decisions in an environment with state transitions and cannot be described by standard normal form games, as they allow only stationary, possibly stochastic, reward functions that depend solely

on the actions of the players. In a normal form game there is no concept of a system with state transitions, a central issue of the Markov decision process concept. Therefore, we now consider a richer framework which generalizes both repeated games and MDPs. Introducing multiple agents to the MDP model significantly complicates the problem that the learning agents face. Both rewards and transitions in the environment now depend on the actions of all agents present in the system. Agents are therefore required to learn in a joint action space. Moreover, since agents can have different goals, an optimal solution which maximizes rewards for all agents simultaneously may fail to exist.

To accommodate the increased complexity of this problem we use the representation of Stochastic of Markov games (Shapley, 1953). While they were originally introduced in game theory as an extension of normal form games, Markov games also generalize the Markov Decision process and were more recently proposed as the standard framework for multi-agent reinforcement learning (Littman, 1994). As the name implies, Markov games still assume that state transitions are Markovian, however, both transition probabilities and expected rewards now depend on the joint action of all agents. Markov games can be seen as an extension of MDPs to the multi-agent case, and of repeated games to multiple state case. If we assume only 1 agent, or the case where other agents play a fixed policy, the Markov game reduces to an MDP. When the Markov game has only 1 state, it reduces to a repeated normal form game.

14.3.1 Markov Games

An extension of the single agent Markov decision process (MDP) to the multi-agent case can be defined by Markov Games. In a Markov Game, joint actions are the result of multiple agents choosing an action independently.

Definition 14.4. A Markov game is a tuple $(n,S,A_{1,\ldots,n},R_{1,\ldots,n},T)$:

- n the number of agents in the system.
- $S = \{s^1,\ldots,s^N\}$ a finite set of system states.
- A_k the action set of agent k.
- $R_k : S \times A_1 \times \ldots \times A_n \times S \to \mathbb{R}$, the reward function of agent k. [1]
- $T : S \times A_1 \times \ldots \times A_n \to \mu(S)$ the transition function.

Note that $A_k(s^i)$ is now the action set available to agent k in state s^i, with $k : 1 \ldots n$, n being the number of agents in the system and $i : 1,\ldots,N$, N being yhe number of states in the system. Transition probabilities $T(s^i,\mathbf{a}^i,s^j)$ and rewards $R^k(s^i,\mathbf{a}^i,s^j)$ now depend on a current state s^i, next state s^j and a joint action from state s^i, i.e. $\mathbf{a}^i = (a^i_1,\ldots a^i_n)$ with $a^i_k \in A_k(s^i)$. The reward function $R_k(s^i,\mathbf{a}^i,s^j,)$ is now individual

[1] As was the case for MDPs, one can also consider the equivalent case where reward does not depend on the next state.

to each agent k. Different agents can receive different rewards for the same state transition. Transitions in the game are again assumed to obey the Markov property.

As was the case in MDPs, agents try to optimize some measure of their future expected rewards. Typically they try to maximize either their future discounted reward or their average reward over time. The main difference with respect to single agent RL, is that now these criteria also depend on the policies of other agents. This results in the following definition for the expected discounted reward for agent k under a joint policy $\pi = (\pi_1, \ldots, \pi_n)$, which assigns a policy π_i to each agent i:

$$V_k^{\pi}(s) = E^{\pi}\left\{ \sum_{t=0}^{\infty} \gamma^t r_k(t+1) \mid s(0) = s \right\} \tag{14.9}$$

while the average reward for agent k under this joint policy is defined as:

$$J_k^{\pi}(s) = \lim_{T\to\infty} \frac{1}{T} E^{\pi}\left\{ \sum_{t=0}^{T} r_k(t+1) \mid s(0) = s \right\} \tag{14.10}$$

Since it is in general impossible to maximize this criterion for all agents simultaneously, as agents can have conflicting goals, agents playing a Markov game face the same coordination problems as in repeated games. Therefore, typically one relies again on equilibria as the solution concept for these problems. The best response and Nash equilibrium concepts can be extended to Markov games, by defining a policy π_k as a best response, when no other policy for agent k exists which gives a higher expected future reward, provided that the other agents keep their policies fixed.

It should be noted that learning in a Markov game introduces several new issues over learning in MDPs with regard to the policy being learned. In an MDP, it is possible to prove that, given some basic assumptions, an optimal deterministic policy always exists. This means it is sufficient to consider only those policies which deterministically map each state to an action. In Markov games, however, where we must consider equilibria between agent policies, this no longer holds. Similarly to the situation in repeated games, it is possible that a discounted Markov game, only has Nash equilibria in which stochastic policies are involved. As such, it is not sufficient to let agents map a fixed action to each state: they must be able to learn a mixed strategy. The situation becomes even harder when considering other reward criteria, such as the average reward, since then it is possible that no equilibria in stationary strategies exist (Gillette, 1957). This means that in order to achieve an equilibrium outcome, the agents must be able to express policies which condition the action selection in a state on the entire history of the learning process. Fortunately, one can introduce some additional assumptions on the structure of the problem to ensure the existence of stationary equilibria (Sobel, 1971).

14.3.2 Reinforcement Learning in Markov Games

While in normal form games the challenges for reinforcement learners originate mainly from the interactions between the agents, in Markov games they face the

additional challenge of an environment with state transitions. This means that the agents typically need to combine coordination methods or equilibrium solvers used in repeated games with MDP approaches from single-agent RL.

14.3.2.1 Value Iteration

A number of approaches have been developed, aiming at extending the successful Q-learning algorithm to multi-agent systems. In order to be successful in a multi-agent context, these algorithms must first deal with a number of key issues.

Firstly, immediate rewards as well as the transition probabilities depend on the actions of all agents. Therefore, in a multi-agent Q-learning approach, the agent does not simply learns to estimate $Q(s,a)$ for each state action pair, but rather estimates $Q(s,\mathbf{a})$ giving the expected future reward for taking the joint action $\mathbf{a} = a_1,\ldots,a_n$ in state s. As such, contrary to the single agent case, the agent does not have a single estimate for the future reward it will receive for taking an action a_k in state s. Instead, it keeps a vector of estimates, which give the future reward of action a_k, depending on the joint action $\mathbf{a}_{-k}$ played by the other agents. During learning, the agent selects an action and then needs to observe the actions taken by other agents, in order to update the appropriate $Q(s,\mathbf{a})$ value.

This brings us to the second issue that a multi-agent Q-learner needs to deal with: the state values used in the bootstrapping update. In the update rule of single agent Q-learning the agent uses a maximum over its actions in the next state s'. This gives the current estimate of the value of state s' under the greedy policy. But as was mentioned above, the agent cannot predict the value of taking an action in the next state, since this value also depends on the actions of the other agents. To deal with this problem, a number of different approaches have been developed which calculate a value of state s' by also taking into account the other agents. All these algorithms, of which we describe a few examples below, correspond to the general multi-agent Q-learning template given in Algorithm 23, though each algorithm differs in the method used to calculate the $V_k(s')$ term in the Q-learning update.

A first possibility to determine the expected value of a state $V_k(s)$ is to employ *opponent modeling*. If the learning agent is able to estimate the policies used by the other agents, it can use this information to determine the expected probabilities with which the different joint actions are played. Based on these probabilities the agent can then determine the expected value of a state. This is the approach followed, for example, by the Joint Action Learner (JAL) algorithm (Claus and Boutilier, 1998). A joint action learner keeps counts $c(s,\mathbf{a}_{-\mathbf{k}})$ of the number of times each state joint action pair $(s,\mathbf{a}_{-\mathbf{k}})$ with $\mathbf{a}_{-\mathbf{k}} \in \mathbb{A}_{-k}$ is played. This information can then be used to determine the empirical frequency of play for the possible joint actions of the other agents:

$$F(s,\mathbf{a}_{-\mathbf{k}}) = \frac{c(s,\mathbf{a}_{-\mathbf{k}})}{\sum_{\mathbf{a}_{-\mathbf{k}}' \in \mathbb{A}_{-k}} n(s,\mathbf{a}_{-\mathbf{k}}')},$$

t=0
Q_k(s,a) = 0 $\forall s,a,k$
repeat
  for all agents k **do**
    select action a_k(t)
  execute joint action $\mathbf{a} = (a_1, \ldots, a_n)$
  observe new state s', rewards r_k
  for all agents k **do**
    $Q_k(\mathrm{s},\mathbf{a}) = Q_k(\mathrm{s},\mathbf{a}) + \alpha\,[R_k(\mathrm{s},\mathbf{a}) + \gamma V_k(s') - Q_k(\mathrm{s},\mathbf{a})]$
until Termination Condition

Algorithm 23. Multi-Agent Q-Learning

This estimated frequency of play for the other agents, allows the joint action learner to calculate the expected Q-values for a state:

$$V_k(s) = \max_{a_k} Q(s,a_k) = \sum_{\mathbf{a}_{-k} \in \mathbb{A}_{-k}} F(s,\mathbf{a}_{-k}).Q(s,a_k,\mathbf{a}_{-k}),$$

where $Q(s,a_k,\mathbf{a}_{-k})$ denotes the Q-value in state s for the joint action in which agent k plays a_k and the other agents play according to $\mathbf{a}_{-k}$. These expected Q-values can then be used for the agent's action selection, as well as in the Q-learning update, just as in the standard single-agent Q-learning algorithm.

Another method used in multi-agent Q-learning is to assume that the other agents will play according to some strategy. For example, in the minimax Q-learning algorithm (Littman, 1994), which was developed for 2-agent zero-sum problems, the learning agent assumes that its opponent will play the action which minimizes the learner's payoff. This means that the max operator of single agent Q-learning is replaced by the minimax value:

$$V_k(s) = min_{a'} max_{\sigma \in \mu(A)} \sum_{a \in A} \sigma(a) Q(s,a,a')$$

The Q-learning agent maximizes over its strategies for state s, while assuming that the opponent will pick the action which minimizes the learner's future rewards. Note that the agent does not just maximizes over the deterministic strategies, as it is possible that the maximum will require a mixed strategy. This system was later generalized to *friend-or-foe* Q-learning (Littman, 2001a), where the learning agent deals with multiple agents by marking them either as friends, who assist to maximize its payoff or foes, who try to minimize the payoff.

Alternative approaches assume that the agents will play an equilibrium strategy. For example, Nash-Q (Hu and Wellman, 2003) observes the rewards for all agents and keeps estimates of Q-values not only for the learning agent, but also for all other agents. This allows the learner to represent the joint action selection in each state as a game, where the entries in the payoff matrix are defined by the Q-values of the agents for the joint action. This representation is also called the *stage game*.

A Nash-Q agent then assumes that all agents will play according to a Nash equilibrium of this stage game in each state:

$$V_k(s) = Nash_k(s,Q_1,...,Q_n),$$

where $Nash_k(s,Q_1,...,Q_n)$ denotes the expected payoff for agent k when the agents play a Nash equilibrium in the stage game for state s with Q-values $Q_1,...,Q_n$. Under some rather strict assumptions on the structure of the stage games, Nash-Q can be shown to converge in self-play to a Nash equilibrium between agent policies.

The approach used in Nash-Q can also be combined with other equilibrium concepts, for example correlated equilibria (Greenwald et al, 2003) or the Stackelberg equilibrium (Kononen, 2003). The main difficulty with these approaches is that the value is not uniquely defined when multiple equilibria exist, and coordination is needed to agree on the same equilibrium. In these cases, additional mechanisms are typically required to select some equilibrium.

While the intensive research into value iteration based multi-agent RL has yielded some theoretical guarantees (Littman, 2001b), convergence results in the general Markov game case remain elusive. Moreover, recent research indicates that a reliance on Q-values alone may not be sufficient to learn an equilibrium policy in arbitrary general sum games (Zinkevich et al, 2006) and new approaches are needed.

14.3.2.2 Policy Iteration

In this section we describe policy iteration for multi-agent reinforcement learning. We focus on an algorithm called Interconnected Learning Automata for Markov Games (MG-ILA)(Vrancx et al, 2008b), based on the learning automata from Section 14.2.2.3 and which can be applied to average reward Markov games. The algorithm can be seen as an implementation of the *actor-critic* framework, where the policy is stored using learning automata. The main idea is straightforward: each agent k puts a single learning automaton LA (k,i) in each system state s^i. At each time step only the automata of the current state are active. Each automaton then individually selects an action for its corresponding agent. The resulting joint action triggers the next state transition and immediate rewards. Automata are not updated using immediate rewards but rather using a response estimating the average reward. The complete algorithm is listed in Algorithm 24.

An interesting aspect of this algorithm is that its limiting behavior can be approximated by considering a normal form game in which all the automata are players. A play in this game selects an action for each agent in each state, and as such corresponds to a pure, joint policy for all agents. Rewards in the game are the expected average rewards for the corresponding joint policies. In (Vrancx et al, 2008b),it is shown that the algorithm will converge to a pure Nash equilibrium in this resulting game (if it exists), and that this equilibrium corresponds to a pure equilibrium between the agent policies. The game approximation also enables an evolutionary game theoretic analysis of the learning dynamics (Vrancx et al, 2008a), similar to that applied to repeated games.

initialise $r_{prev}(s,k)$, $t_{prev}(s)$, $a_{prev}(s,k)$,t, $r_{tot}(k)$,$\rho_k(s,a)$, $\eta_k(s,a)$ to zero, $\forall s,k,a$.
$s \leftarrow s(0)$
loop
for all Agents k **do**
if s was visited before **then**
• Calculate received reward and time passed since last visit to state s:

$$\Delta r_k = r_{tot}(k) - r_{prev}(s,k)$$

$$\Delta t = t - t_{prev}(s)$$

• Update estimates for action $a_{prev}(s,k)$ taken on last visit to s:

$$\rho_k(s,a_{prev}(s,k)) = \rho_k(s,a_{prev}(s,k)) + \Delta r_k$$

$$\eta_k(s,a_{prev}(s,k)) = \eta_k(s,a_{prev}(s,k)) + \Delta t$$

• Calculate feedback:

$$\beta_k(t) = \frac{\rho_k(s,a_{prev}(s,k))}{\eta_k(s,a_{prev}(s,k))}$$

• Update automaton $LA(s,k)$ using L_{R-I} update with $a(t) = a_{prev}(s,k)$ and $\beta_k(t)$ as above.
• Let $LA(s,k)$ select an action a_k.
• Store data for current state visit:

$$t_{prev}(s) \leftarrow t$$

$$r_{prev}(s,k) \leftarrow r_{tot}(k)$$

$$a_{prev}(s,k) \leftarrow a_k$$

• Execute joint action $\mathbf{a} = (a_1,\ldots,a_n)$, observe immediate rewards r_k and new state s'
• $s \leftarrow s'$
• $r_{tot}(k) \leftarrow r_{tot}(k) + r_k$
• $t \leftarrow t+1$

Algorithm 24. MG-ILA

While not as prevalent as value iteration based methods, a number of interesting approaches based on policy iteration have been proposed. Like the algorithm described above, these methods typically rely on a gradient based search of the policy space. (Bowling and Veloso, 2002), for example, proposes an actor-critic framework which combines tile coding generalization with policy gradient ascent and uses the Win or Learn Fast (WoLF) principle. The resulting algorithm is empirically shown to learn in otherwise intractably large problems. (Kononen, 2004) introduces a policy gradient method for common-interest Markov games which extends the single agent methods proposed by (Sutton et al, 2000). Finally, (Peshkin et al, 2000)

develop a gradient based policy search method for partially observable, identical payoff stochastic games. The method is shown to converge to local optima which are, however, not necessarily Nash equilibria between agent policies.

14.4 Sparse Interactions in Multi-agent System

A big drawback of reinforcement learning in Markov games is the size of the state-action-space in which the agents are learning. All agents learn in the entire joint state-action space and as such these approaches become quickly infeasible for all but the smallest environments and with a limited number of agents. Recently, a lot of attention has gone into mitigating this problem. The main intuition for these approaches is to only explicitly consider the other agents if a better payoff can be obtained by doing so. In all other situations the other agents can safely be ignored and as such have the advantages of learning in a small state-action space, while also having access to the necessary information to deal with the presence of other agents, if this is beneficial. An example of such systems is an automated warehouse, where the automated guided vehicles only have to consider each other when they are close by to each other. We can distinguish two different lines of research: agents can base their decision for coordination on the actions that are selected, or agents can focus on the state information at their disposal, and learn when it is beneficial to observe the state information of other agents. We will describe both these approaches separately in Sections 14.4.2.1 and 14.4.2.2

14.4.1 Learning on Multiple Levels

Learning with sparse interactions provides an easy way of dealing with the exponential growth of the state space in terms of the number of agents involved in the learning process. Agents should only rely on more global information, in those situations where the transition of the state of the agent to the next state and the rewards the agents experience are not only dependent on the local state information of the agent performing the action, but also on the state information or actions of other agents. The idea of sparse interactions is *'When is an agent experiencing influence from another agent?'*. Answering this questing, allows an agent to know when it can select its actions independently (i.e. the state transition function and reward function are only dependent on its own action) or when it must coordinate with other agents (i.e. the state transition function and the reward function is the effect of the joint action of multiple agents). This leads naturally to a decomposition of the multi-agent learning process into two separate layers. The top layer should learn when it is necessary to observe the state information about other agents and select whether pure independent learning is sufficient, or whether some form of coordination between the agents is required. The bottom layer contains a single agent learning technique, to be used when there is no risk of influence by other agents, and a multi-agent technique, to be used when the state transition and reward the agent receives is dependent of the current state and actions of other agents. Figure 14.2 shows a graphical

representation of this framework. In the following subsection we begin with an overview of algorithms that approach this problem from the action space point of view, and focus on the coordination of actions.

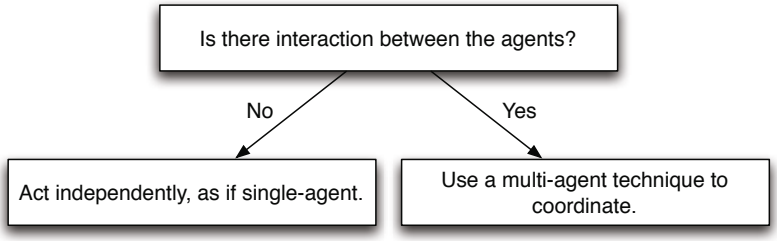

Fig. 14.2 Decoupling the learning process by learning when to take the other agent into account on one level, and acting on the second level

14.4.2 Learning to Coordinate with Sparse Interactions

14.4.2.1 Learning Interdependencies among Agents

Kok & Vlassis proposed an approach based on a sparse representation for the joint action space of the agents while observing the entire joint state space. More specifically they are interested in learning joint-action values for those states where the agents explicitly need to coordinate. In many problems, this need only occurs in very specific situations (Guestrin et al, 2002b). *Sparse Tabular Multiagent Q-learning* maintains a list of states in which coordination is necessary. In these states, agents select a joint action, whereas in all the uncoordinated states they all select an action individually (Kok and Vlassis, 2004b). By replacing this list of states by coordination graphs (CG) it is possible to represent dependencies that are limited to a few agents (Guestrin et al, 2002a; Kok and Vlassis, 2004a, 2006). This technique is known as *Sparse Cooperative Q-learning* (SCQ). Figure 14.3 shows a graphical representation of a simple CG for a situation where the effect of the actions of agent 4 depend on the actions of agent 2 and the actions of agent 2 and 3 both depend on the actions of agent 1, so the nodes represent the agents, while an edge defines a dependency between two agents. If agents transitioned into a coordinated state, they applied a variable elimination algorithm to compute the optimal joint action for the current state. In all other states, the agents select their actions independently.

In later work, the authors introduced *Utile Coordination* (Kok et al, 2005). This is a more advanced algorithm that uses the same idea as SCQ, but instead of having to define the CGs beforehand, they are being learned online. This is done by maintaining statistical information about the obtained rewards conditioned on the states and actions of the other agents. As such, it is possible to learn the context specific dependencies that exist between the agents and represent them in a CG. This technique is however limited to fully cooperative MAS.

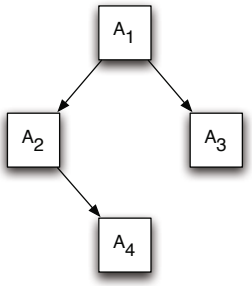

Fig. 14.3 Simple coordination graph. In the situation depicted the effect of the actions of agent 4 depends on the actions of agent 2 and the actions of agent 2 and 3 both depend on the actions of agent 1.

The primary goal of these approaches is to reduce the joint-action space. However, the computation or learning in the algorithms described above, always employ a complete multi-agent view of the entire joint-state space to select their actions, even in states where only using local state information would be sufficient. As such, the state space in which they are learning is still exponential in the number of agents, and its use is limited to situations in which it is possible to observe the entire joint state.

14.4.2.2 Learning a Richer State Space Representation

Instead of explicitly learning the optimal coordinated action, a different approach consists in learning in which states of the environment it is beneficial to include the state information about other agents. We will describe two different methods. The first method learns in which states coordination is beneficial using an RL approach. The second method learns the set of states in which coordination is necessary based on the observed rewards. Unlike the approaches mentioned in Section 14.4.2.1, these approaches can also be applied to conflicting interest games and allow independent action selection.

The general idea of the approaches described in this section are given by Figure 14.4. These algorithms will expand the local state information of an agent to incorporate the information of another agent if this information is necessary to avoid suboptimal rewards.

Learning of Coordination

Spaan and Melo approached the problem of coordination from a different angle than Kok & Vlassis (Spaan and Melo, 2008). They introduced a new model for multi-agent decision making under uncertainty called *interaction-driven Markov games* (IDMG). This model contains a set of interaction states which lists all the states in which coordination is beneficial.

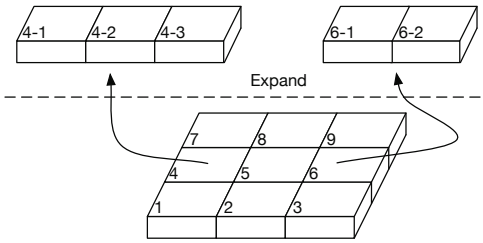

Fig. 14.4 Graphical representation of state expansion with sparse interactions. Independent single states are expanded to joint-states where necessary. Agents begin with 9 independent states. After a while states 4 and 6 of an agent are expanded to include the states of another agent.

In later work, Melo and Veloso introduced an algorithm where agents learn in which states they need to condition their actions on the local state information of other agents (Melo and Veloso, 2009). As such, their approach can be seen as a way of solving an IDMG where the states in which coordination is necessary is not specified beforehand. To achieve this, they augment the action space of each agent with a pseudo-coordination action (`COORDINATE`). This action will perform an active perception step. This could for instance be a broadcast to the agents to divulge their location or using a camera or sensors to detect the location of the other agents. This active perception step will decide whether coordination is necessary or if it is safe to ignore the other agents. Since the penalty of miscoordination is bigger than the cost of using the active perception, the agents learn to take this action in the interaction states of the underlying IDMG. This approach solves the coordination problem by deferring it to the active perception mechanism.

The active perception step of LoC can consist of the use of a camera, sensory data, or communication to reveal the local state information of another agent. As such the outcome of the algorithm depends on the outcome of this function. Given an adequate active perception function, LoC is capable of learning a sparse set of states in which coordination should occur. Note that depending on the active perception function, this algorithm can be used for both cooperative as conflicting interest systems.

The authors use a variation on the standard Q-learning update rule:

$$Q_k^C(s,a_k) \leftarrow (1-\alpha(t))Q_k^C(s,a) + \alpha(t)\left[r_k + \gamma \max_{a'} Q_k(s'_k,a'_k)\right] \tag{14.11}$$

Where Q_k^C represents the Q-table containing states in which agent k will coordinate and Q_k contains the state-action values for its independent states. The joint state information is represented as s, whereas s_k and a_k are the local state information and action of agent k. So the update of Q_k^C uses the estimates of Q_k. This represents the one-step behaviour of the `COORDINATE` action and allows for a sparse representation of Q_k^C, since there is no direct dependency between the states in this joint Q-table.

Coordinating Q-Learning

Coordinating Q-Learning, or CQ-learning, learns in which states an agent should take the other agents into consideration (De Hauwere et al, 2010) and in which states is cant act using primarily only its own state information. More precisely, the algorithm will identify states in which an agent should take other agents into account when choosing its preferred action.

The algorithm can be decomposed into three sections: detecting conflict situations, selecting actions and updating the Q-values which will now be explained in more detail:

1. **Detecting conflict situations**
   Agents must identify in which states they experience the influence of at least one other agent. CQ-Learning needs a baseline for this, so agents are assumed to have learned a model about the expected payoffs for selecting an action in a particular state applying an individual policy. For example, in a gridworld this would mean that the agents have learned a policy to reach some goal, while being the only agent present in the environment. If agents are influencing each other, this will be reflected in the payoff the agents receive when they are acting together. CQ-learning uses a statistical test to detect if there are changes in the observed rewards for the selected state-action pair compared to the case where they were acting alone in the environment. Two situations can occur:

   a. The statistics allow to detect a change in the received immediate rewards. In this situation, the algorithm will mark this state, and search for the cause of this change by collecting new samples from the joint state space in order to identify the joint state-action pairs in which collisions occur. These state-action pairs are then marked as being *dangerous*, and the state space of the agent is augmented by adding this joint state information. State-action pairs that did not cause interactions are marked as being *safe*, i.e. the agent's actions in this state are independent from the states of other agents. So the algorithm will first attempt to detect changes in the rewards an agent receives, solely based on its own state, before trying to identify due to which other agents these changes occur.
   b. The statistics indicate that the rewards the agent receives are from the same distribution as if the agent was acting alone. Therefore, no special action is taken in this situation and the agent continues to act as if it was alone.

2. **Selecting actions**
   If an agent selects an action, it will check if its current local state is a state in which a discrepancy has been detected previously (case 1.a, described above). If so, it will observe the global state information to determine if the state information of the other agents is the same as when the conflict was detected. If this is the case, it will condition its actions on this global state information, otherwise it can act independently, using only its own local state information. If its local state information has never caused a discrepancy (case 1.b, described above), it can act without taking the other agents into consideration.

3. **Updating the Q-values**
   The updating the Q-values follows the same idea as the Learning of Coordination algorithm, described above. The Q-values for local states are used to bootstrap the Q-values of the states that were augmented.

The statistical test used in the algorithm is the Student t-test (Stevens, J.P., 1990). This test can determine whether the null hypothesis that the mean of two populations of samples are equal holds, against the alternative that they are not equal. In CQ-learning this test is first used to identify in which states the observed rewards are significantly different from the expected rewards based on single agent learning, and also to determine on which other agents' states these changes depend.

A formal description of this algorithm is given in Algorithm 25.

CQ-learning can also be used to generalise information from states in which coordination is necessary to obtain a state-independent representation of the coordination dependencies that exist between the agents (De Hauwere et al, 2010). This information can then be transferred to other, more complex, task environments (Vrancx et al, 2011). This principle of transfer learning improves the learning speed, since agents can purely focus on the core task of the problem at hand and use transferred experience for the coordination issues.

Initialise Q_k through single agent learning and Q_k^j;
while true do
  if state s_k of Agent k is unmarked **then**
    Select a_k for Agent k from Q_k
  else
    if the joint state information js is safe **then**
      Select a_k for Agent k from Q_k
    else
      Select a_k for Agent k from Q_k^j based on the joint state information js
  Sample $\langle s_k, a_k, r_k \rangle$
  if t-test detects difference in observed rewards vs expected rewards for $\langle s_k, a_k \rangle$ **then**
    mark s_k
    for $\forall$ other state information present in the joint state js **do**
      if t-test detects difference between independent state s_k and joint state js **then**
        add js to Q_k^j
        mark js as dangerous
      else
        mark js as safe
  if s_k is unmarked for Agent k or js is safe **then**
    No need to update $Q_k(s_k)$.
  else
    Update $Q_k^j(js, a_k) \leftarrow (1-\alpha_t) Q_k^j(js, a_k) + \alpha_t [r(js, a_k) + \gamma \, max_a Q(s'_k, a)]$

Algorithm 25. CQ-Learning algorithm for agent k

This approach was later extended to detect sparse interactions that are only reflected in the reward signal, several timesteps in the future (De Hauwere et al, 2011). Examples of such situations are for instance if the order in which goods arrive in a warehouse are important.

14.5 Further Reading

Multi-agent reinforcement learning is a growing field of research, but quite some challenging research questions are still open. A lot of the work done in single-agent reinforcement learning, such as the work done in Bayesian RL, batch learning or transfer learning, has yet to find its way to the multi-agent learning community. General overviews of multi-agent systems are given by Weiss (Weiss, G., 1999), Wooldridge (Wooldridge, M., 2002) and more recently Shoham (Shoham, Y. and Leyton-Brown, K., 2009). For a thorough overview of the field of Game Theory the book by Gintis will be very useful (Gintis, H., 2000).

More focused on the domain of multi-agent reinforcement learning we recommend the paper by Buşoniu which gives an extensive overview of recent research and describes a representative selection of MARL algorithms in detail together with their strengths and weaknesses (Busoniu et al, 2008). Another track of multi-agent research considers systems where agents are not aware of the type or capabilities of the other agents in the system (Chalkiadakis and Boutilier, 2008).

An important issue in multi-agent reinforcement learning as well as in single agent reinforcement learning, is the fact that the reward signal can be delayed in time. This typically happens in systems which include queues, like for instance in network routing and job scheduling. The immediate feedback of taking an action can only be provided to the agent after the effect of the action becomes apparent, e.g. after the job is processed. In (Verbeeck et al, 2005) a policy iteration approach based on learning automata is given, which is robust with respect to this type of delayed reward. In (Littman and Boyan, 1993) a value iteration based algorithm is described for routing in networks. An improved version of the algorithm is presented in (Steenhaut et al, 1997; Fakir, 2004). The improved version allows on one hand to use the feedback information in a more efficient way, and on the other hand it avoids instabilities that might occur due to careless exploration.

References

Aumann, R.: Subjectivity and Correlation in Randomized Strategies. Journal of Mathematical Economics 1(1), 67–96 (1974)

Bowling, M.: Convergence and No-Regret in Multiagent Learning. In: Advances in Neural Information Processing Systems 17 (NIPS), pp. 209–216 (2005)

Bowling, M., Veloso, M.: Convergence of Gradient Dynamics with a Variable Learning Rate. In: Proceedings of the Eighteenth International Conference on Machine Learning (ICML), pp. 27–34 (2001)

Bowling, M., Veloso, M.: Scalable Learning in Stochastic Games. In: AAAI Workshop on Game Theoretic and Decision Theoretic Agents (2002)

Busoniu, L., Babuska, R., De Schutter, B.: A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 38(2), 156–172 (2008)

Chalkiadakis, G., Boutilier, C.: Sequential Decision Making in Repeated Coalition Formation under Uncertainty. In: Parkes, P.M., Parsons (eds.) Proceedings of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008), pp. 347–354 (2008)

Claus, C., Boutilier, C.: The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems. In: Proceedings of the National Conference on Artificial Intelligence, pp. 746–752. John Wiley & Sons Ltd. (1998)

De Hauwere, Y.M., Vrancx, P., Nowé, A.: Learning Multi-Agent State Space Representations. In: Proceedings of the 9th International Conference on Autonomous Agents and Multi-Agent Systems, Toronto, Canada, pp. 715–722 (2010)

De Hauwere, Y.M., Vrancx, P., Nowé, A.: Detecting and Solving Future Multi-Agent Interactions. In: Proceedings of the AAMAS Workshop on Adaptive and Learning Agents, Taipei, Taiwan, pp. 45–52 (2011)

Dorigo, M., Stützle, T.: Ant Colony Optimization. Bradford Company, MA (2004)

Fakir, M.: Resource Optimization Methods for Telecommunication Networks. PhD thesis, Department of Electronics and Informatics, Vrije Universiteit Brussel, Belgium (2004)

Foster, D., Young, H.: Regret Testing: A Simple Payoff-based Procedure for Learning Nash Equilibrium. University of Pennsylvania and Johns Hopkins University, Mimeo (2003)

Gillette, D.: Stochastic Games with Zero Stop Probabilities. Ann. Math. Stud. 39, 178–187 (1957)

Gintis, H.: Game Theory Evolving. Princeton University Press (2000)

Greenwald, A., Hall, K., Serrano, R.: Correlated Q-learning. In: Proceedings of the Twentieth International Conference on Machine Learning, pp. 242–249 (2003)

Guestrin, C., Lagoudakis, M., Parr, R.: Coordinated Reinforcement Learning. In: Proceedings of the 19th International Conference on Machine Learning, pp. 227–234 (2002a)

Guestrin, C., Venkataraman, S., Koller, D.: Context-Specific Multiagent Coordination and Planning with Factored MDPs. In: 18th National Conference on Artificial Intelligence, pp. 253–259. American Association for Artificial Intelligence, Menlo Park (2002b)

Hart, S., Mas-Colell, A.: A Reinforcement Procedure Leading to Correlated Equilibrium. Economic Essays: A Festschrift for Werner Hildenbrand, 181–200 (2001)

Hu, J., Wellman, M.: Nash Q-learning for General-Sum Stochastic Games. The Journal of Machine Learning Research 4, 1039–1069 (2003)

Kapetanakis, S., Kudenko, D.: Reinforcement Learning of Coordination in Cooperative Multi-Agent Systems. In: Proceedings of the National Conference on Artificial Intelligence, pp. 326–331. AAAI Press, MIT Press, Menlo Park, Cambridge (2002)

Kapetanakis, S., Kudenko, D., Strens, M.: Learning to Coordinate Using Commitment Sequences in Cooperative Multiagent-Systems. In: Proceedings of the Third Symposium on Adaptive Agents and Multi-agent Systems (AAMAS-2003), p. 2004 (2003)

Kok, J., Vlassis, N.: Sparse Cooperative Q-learning. In: Proceedings of the 21st International Conference on Machine Learning. ACM, New York (2004a)

Kok, J., Vlassis, N.: Sparse Tabular Multiagent Q-learning. In: Proceedings of the 13th Benelux Conference on Machine Learning, Benelearn (2004b)

Kok, J., Vlassis, N.: Collaborative Multiagent Reinforcement Learning by Payoff Propagation. Journal of Machine Learning Research 7, 1789–1828 (2006)

Kok, J., 't Hoen, P., Bakker, B., Vlassis, N.: Utile Coordination: Learning Interdependencies among Cooperative Agents. In: Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG 2005), pp. 29–36 (2005)

Kononen, V.: Asymmetric Multiagent Reinforcement Learning. In: IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003), pp. 336–342 (2003)

Könönen, V.: Policy Gradient Method for Team Markov Games. In: Yang, Z.R., Yin, H., Everson, R.M. (eds.) IDEAL 2004. LNCS, vol. 3177, pp. 733–739. Springer, Heidelberg (2004)

Leyton-Brown, K., Shoham, Y.: Essentials of Game Theory: A Concise Multidisciplinary Introduction. Synthesis Lectures on Artificial Intelligence and Machine Learning 2(1), 1–88 (2008)

Littman, M.: Markov Games as a Framework for Multi-Agent Reinforcement Learning. In: Proceedings of the Eleventh International Conference on Machine Learning, pp. 157–163. Morgan Kaufmann (1994)

Littman, M.: Friend-or-Foe Q-learning in General-Sum Games. In: Proceedings of the Eighteenth International Conference on Machine Learning, pp. 322–328. Morgan Kaufmann (2001a)

Littman, M.: Value-function Reinforcement Learning in Markov Games. Cognitive Systems Research 2(1), 55–66 (2001b), http://www.sciencedirect.com/science/article/B6W6C-430G1TK-4/2/822caf1574be32ae91adf15de90becc4, doi:10.1016/S1389-0417(01)00015-8

Littman, M., Boyan, J.: A Distributed Reinforcement Learning Scheme for Network Routing. In: Proceedings of the 1993 International Workshop on Applications of Neural Networks to Telecommunications, pp. 45–51. Erlbaum (1993)

Mariano, C., Morales, E.: DQL: A New Updating Strategy for Reinforcement Learning Based on Q-Learning. In: Flach, P.A., De Raedt, L. (eds.) ECML 2001. LNCS (LNAI), vol. 2167, pp. 324–335. Springer, Heidelberg (2001)

Melo, F., Veloso, M.: Learning of Coordination: Exploiting Sparse Interactions in Multiagent Systems. In: Proceedings of the 8th International Conference on Autonomous Agents and Multi-Agent Systems, pp. 773–780 (2009)

Nash, J.: Equilibrium Points in n-Person Games. Proceedings of the National Academy of Sciences of the United States of America, 48–49 (1950)

Peshkin, L., Kim, K., Meuleau, N., Kaelbling, L.: Learning to Cooperate via Policy Search. In: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, UAI 2000, pp. 489–496. Morgan Kaufmann Publishers Inc., San Francisco (2000), http://portal.acm.org/citation.cfm?id=647234.719893

Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice-Hall, Englewood Cliffs (2003)

Sastry, P., Phansalkar, V., Thathachar, M.: Decentralized Learning of Nash Equilibria in Multi-Person Stochastic Games with Incomplete Information. IEEE Transactions on Systems, Man and Cybernetics 24(5), 769–777 (1994)

Shapley, L.: Stochastic Games. Proceedings of the National Academy of Sciences 39(10), 1095–1100 (1953)

Shoham, Y., Leyton-Brown, K.: Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press (2009)

Singh, S., Kearns, M., Mansour, Y.: Nash Convergence of Gradient Dynamics in General-Sum Games. In: Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence, pp. 541–548 (2000)

Smith, J.: Evolution and the Theory of Games. Cambridge Univ. Press (1982)

Sobel, M.: Noncooperative Stochastic Games. The Annals of Mathematical Statistics 42(6), 1930–1935 (1971)

Spaan, M., Melo, F.: Interaction-Driven Markov Games for Decentralized Multiagent Planning under Uncertainty. In: Proceedings of the 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 525–532. International Foundation for Autonomous Agents and Multiagent Systems (2008)

Steenhaut, K., Nowe, A., Fakir, M., Dirkx, E.: Towards a Hardware Implementation of Reinforcement Learning for Call Admission Control in Networks for Integrated Services. In: Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications, vol. 3, p. 63. Lawrence Erlbaum (1997)

Stevens, J.P.: Intermediate Statistics: A Modern Approach. Lawrence Erlbaum (1990)

Sutton, R., McAllester, D., Singh, S., Mansour, Y.: Policy Gradient Methods for Reinforcement Learning with Function Approximation. In: Advances in Neural Information Processing Systems, vol. 12(22) (2000)

Tsitsiklis, J.: Asynchronous stochastic approximation and Q-learning. Machine Learning 16(3), 185–202 (1994)

Tuyls, K., Nowé, A.: Evolutionary Game Theory and Multi-Agent Reinforcement Learning. The Knowledge Engineering Review 20(01), 63–90 (2005)

Verbeeck, K.: Coordinated Exploration in Multi-Agent Reinforcement Learning. PhD thesis, Computational Modeling Lab, Vrije Universiteit Brussel, Belgium (2004)

Verbeeck, K., Nowe, A., Tuyls, K.: Coordinated Exploration in Multi-Agent Reinforcement Learning: An Application to Loadbalancing. In: Proceedings of the 4th International Conference on Autonomous Agents and Multi-Agent Systems (2005)

Vrancx, P., Tuyls, K., Westra, R.: Switching Dynamics of Multi-Agent Learning. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), vol. 1, pp. 307–313. International Foundation for Autonomous Agents and Multiagent Systems, Richland (2008a), http://portal.acm.org/citation.cfm?id=1402383.1402430

Vrancx, P., Verbeeck, K., Nowe, A.: Decentralized Learning in Markov Games. IEEE Transactions on Systems, Man, and Cybernetics, Part B 38(4), 976–981 (2008b)

Vrancx, P., De Hauwere, Y.M., Nowé, A.: Transfer learning for Multi-Agent Coordination. In: Proceedings of the 3th International Conference on Agents and Artificial Intelligence, Rome, Italy, pp. 263–272 (2011)

Weiss, G.: Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence. The MIT Press (1999)

Wheeler Jr., R., Narendra, K.: Decentralized Learning in Finite Markov Chains. IEEE Transactions on Automatic Control 31(6), 519–526 (1986)

Williams, R.: Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. Machine Learning 8(3), 229–256 (1992)

Wooldridge, M.: An Introduction to Multi Agent Systems. John Wiley and Sons Ltd. (2002)

Wunder, M., Littman, M., Babes, M.: Classes of Multiagent Q-learning Dynamics with epsilon-greedy Exploration. In: Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel, pp. 1167–1174 (2010)

Zinkevich, M.: Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In: Machine Learning International Conference, vol. 20(2), p. 928 (2003)

Zinkevich, M., Greenwald, A., Littman, M.: Cyclic equilibria in Markov games. In: Advances in Neural Information Processing Systems, vol. 18, p. 1641 (2006)