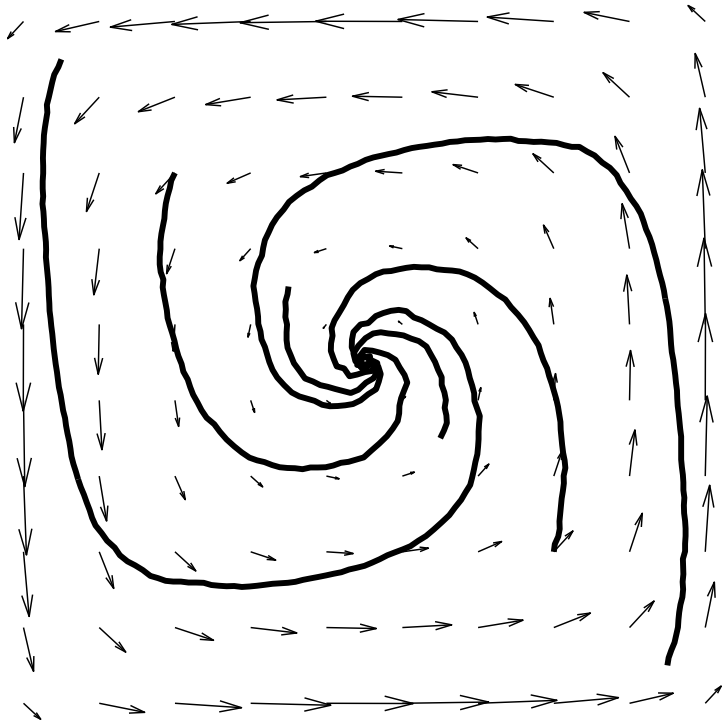


analyzing **Reinforcement Learning** algorithms using **Evolutionary Game Theory**

Daan Bloembergen

ANALYZING REINFORCEMENT LEARNING ALGORITHMS USING EVOLUTIONARY GAME THEORY

Daan Bloembergen

Master Thesis DKE 10-09

Thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Science in Artificial Intelligence
at the Faculty of Humanities and Sciences
of Maastricht University

Thesis committee:

Dr. Karl Tuyls
Michael Kaisers, M.Sc.
Prof.dr. Gerhard Weiss
Dr. Ronald Westra

Maastricht University
Department of Knowledge Engineering
Maastricht, The Netherlands
June 2010

The cover image visualizes several learning trajectories of Lenient Frequency Adjusted Q-learning in the Matching Pennies game, together with the dynamics of its evolutionary model. This image is related to the experiments described in Section 6.1.

Acknowledgments

A master's thesis is not something that suddenly springs into existence. Not completely unrelated to my research, I came to think of it as a dynamical process, sometimes slow moving, sometimes fast, in one direction or another. Only if you know the governing dynamics, will you know what the end result will be. To extend this metaphor further (again not completely unrelated to my research), thesis writing resembles multi-agent interactions, where you are not on your own, but influenced by a group of people around you. Without these influences, this thesis would not have been anything like it is now.

First of all, I would like to express my gratitude to my two supervisors. Karl, thank you for your support, enthusiasm, and constructive comments during our many meetings. There have been periods that my research did not proceed as well as I hoped, or that I lacked motivation or inspiration; but every time one meeting with you was enough to get me back on track. I hope that we get the chance to continue this fruitful cooperation in the future. Michael, thank you for all your good comments and suggestions.

I would also like to thank all my good friends and fellow students at the Department of Knowledge Engineering, for making my time there a very enjoyable one. Furthermore, I thank everyone who supported me in any way during my study, especially my closest friends and family. Without you I would not have been where I am now.

As an old saying goes, "the proof of the pudding is in the eating". Therefore, I hope that you will enjoy reading this thesis as much as I enjoyed the process of writing it.

Summary

Multi-agent learning plays an increasingly important role in solving complex dynamic problems in today's society. Despite its importance, the field of multi-agent learning still lacks a strong theoretical framework. Recently, an evolutionary game theoretic approach to multi-agent reinforcement learning has been proposed as a first step towards such a framework. This thesis contributes by investigating how a better understanding of multi-agent reinforcement learning can be derived from the evolutionary game theoretic analysis, both in homogeneous and heterogeneous environments.

Simulation experiments are performed in the domain of 2×2 normal form games. The results of these simulations are compared to the evolutionary predictions based on the replicator dynamics. This comparison shows that evolutionary game theory provides an efficient way to predict the behavior, convergence properties and performance of reinforcement learners. These insights can be used to select an appropriate learning algorithm for a given task, or to guide parameter tuning.

Moreover, it is demonstrated how insights from evolutionary game theory can be used to improve the performance of reinforcement learning algorithms. Lenient Frequency Adjusted Q-learning is proposed as a learning algorithm that implements the lenient evolutionary model derived by Panait, Tuyls, and Luke (2008). It is argued theoretically and shown empirically that the proposed learning algorithm indeed matches the dynamical model, and that its resulting behavior is more preferable than the behavior of the original Lenient Q-learning algorithm.

The results achieved in this thesis confirm that the evolutionary game theoretic approach is a promising road to arrive at a general theoretical framework of multi-agent learning. Not only can this approach help to describe and analyze the behavior and performance of reinforcement learning algorithms, it can also point towards possible improvements, and it can even be used to design new learning algorithms.

Contents

Glossary

Abbreviations

EGT	Evolutionary Game Theory
ESS	Evolutionarily Stable Strategy
FALA	Finite Action-set Learning Automata
FAQ	Frequency Adjusted Q-learning
GT	Game Theory
LFAQ	Lenient Frequency Adjusted Q-learning
LQ	Lenient Q-learning
MARL	Multi-Agent Reinforcement Learning
MAS	Multi-Agent System
NE	Nash Equilibrium
PW	Polynomial Weights (regret minimization algorithm)
RD	Replicator Dynamics
RL	Reinforcement Learning
RM	Regret Minimization

Symbols

α	step size parameter
β	step size parameter
γ	discount factor
κ	degree of leniency
λ	step size parameter
τ	temperature parameter of the Boltzmann distribution
a	set of actions
Q	Action-value function of Q-learning
r_i	reward received for playing action a_i
x	probability distribution over actions (policy)

Chapter 1

Introduction

The goal of this thesis is to explore the strengths and benefits of, and to further deepen, the evolutionary game theoretic approach to multi-agent reinforcement learning. This chapter provides an introduction to multi-agent learning, its relation to evolutionary game theory, and the formal link between the two fields that forms the basis of this approach. The contribution of this thesis to both fields is described, and an outline of the thesis is presented.

1.1 Learning in multi-agent systems

Recent years have seen an increasing interest in multi-agent learning within the field of Artificial Intelligence (AI) (Panait and Luke, 2005; Tuyls and Nowé, 2005; Shoham, Powers, and Grenager, 2007; Busoniu, Babuška, and De Schutter, 2008). In a multi-agent system (MAS), several autonomous agents interact in a single environment, thereby possibly influencing each other's behavior. Multi-agent systems provide a framework to describe and solve many complex problems in today's society, leading to a wide range of applications. Examples include, but are certainly not limited to, robotics, scheduling, resource management, information retrieval, and space and air traffic control (Weiss, 1999; Wooldridge, 2002; Busoniu *et al.*, 2008).

Multi-agent systems offer several significant advantages over traditional *centralized* systems (Weiss, 1999). They allow for tasks to be distributed over multiple agents, enabling their parallel execution. This in turn improves the scalability of such systems, as their modular nature allows to easily add or remove parts as the size of the problem changes. Moreover, the fact that each agent works independently makes it easy to built in redundancies, which increases the system's robustness; should one agent fail, another agent takes over and the system can continue to operate properly. These properties make multi-agent systems, compared to centralized systems, a more preferable solution to many real world problems.

The fact that multiple agents interact leads to a highly dynamic, non-deterministic environment. In such an environment, defining proper behavior for each agent in advance is a highly complex task. Therefore, *learning* is essential in most multi-agent systems. One of the most basic types of learning is learning from interaction, i.e., by perceiving the effect that a certain action has on the environment. It is this type of learning that underlies most theories of learning and intelligence (Sutton and Barto, 1998). Three main types of interaction-based learning are usually distinguished. In *supervised learning*, an agent is presented with input-output examples, and has to learn a generalized mapping from input to output based on these examples. *Unsupervised learning*, on the other hand, involves learning patterns in the input without knowing what the correct output should be. The third form of learning is *reinforcement learning* (RL), in which a learner receives a reinforcement signal that tells it something about the quality of its output, without specifying explicitly whether the output was 'correct'. In many complex domains it is impossible to provide exact input-output samples for reliable supervised learning, making reinforcement learning the only feasible learning algorithm in such cases.

Single-agent reinforcement learning has already been studied in much detail and acquired a strong theoretical foundation (Kaelbling, Littman, and Moore, 1996; Sutton and Barto, 1998). This allowed for the construction of proofs of convergence for several RL algorithms, e.g., Q-learning (Watkins and Dayan, 1992). However, despite some specific theoretical proofs of convergence in multi-agent environments (e.g., Bowling and Veloso, 2002), such a general framework is still lacking for multi-agent reinforcement learning.

Recently, an evolutionary game theoretic approach to reinforcement learning has been proposed to bridge this gap.

1.2 The evolutionary game theoretic approach

Learning in multi-agent systems is not only relevant within the field of AI; also in game theory, multi-agent learning has been extensively studied (Shoham *et al.*, 2007). It is not surprising then, that the two fields share a lot of common ground concerning this topic. For example, game theoretic concepts such as Nash equilibria translate easily to the area of multi-agent systems. Indeed, game theory often provides the environment in which multi-agent systems are modeled (Wooldridge, 2002).

Recently, the focus has shifted from traditional game theory to evolutionary game theory (EGT) (Gintis, 2009). The concepts employed by EGT proved well suited to describe learning in multi-agent systems. Both fields are concerned with dynamic environments with a high level of uncertainty, characterized by the fact that agents lack complete information (Tuyls, 't Hoen, and Vanschoenwinkel, 2006). Moreover, there exists a formal relation between EGT and RL, based on the Replicator Dynamics (RD) that describe the change in a population of players over time. This relation was first established by Börgers and Sarin (1997), showing that, in the continuous time limit, Cross learning converges to the replicator dynamics. In the past decade, this formal link has been extended to other forms of reinforcement learning such as Q-learning, Learning Automata, and Regret Minimization (Tuyls *et al.*, 2002; Tuyls, Verbeeck, and Lenaerts, 2003b; Tuyls *et al.*, 2006; Klos, Ahee, and Tuyls, 2010).

This formal relation forms the basis of the evolutionary game theoretic approach to reinforcement learning. It provides new insights towards the understanding, analysis, and design of multi-agent RL algorithms (Tuyls *et al.*, 2006). This approach has several important advantages. First of all, it sheds light into the black box of reinforcement learning, by making it possible to analyze the learning dynamics of RL algorithms in detail. This in turn facilitates important tasks such as parameter tuning. Furthermore, studying the dynamics of different RL algorithms helps in selecting a specific learner for a given problem. The triangular relation between multi-agent systems, evolutionary game theory and reinforcement learning is presented graphically in Figure 1.1, where the open arrow indicates the role of the evolutionary game theoretic approach.

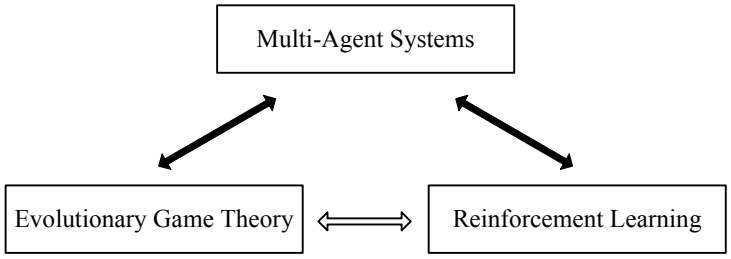


Figure 1.1: The triangular relation between reinforcement learning, multi-agent systems and evolutionary game theory (Tuyls, 2004). RL is a learning method suited to dynamic environments, such as MAS; EGT provides the theoretical means to describe learning in MAS; and the replicator dynamics formally link EGT and RL.

1.3 The contributions of this thesis

This thesis combines two different approaches to reinforcement learning. The 'traditional' approach analyzes RL algorithms by extensive simulation, and focuses mainly on the *performance* of the learners rather than giving a qualitative description of their underlying behavior (e.g., Jafari *et al.*, 2001; Bab and Brafman, 2008; Kalyanakrishnan and Stone, 2009). The evolutionary game theoretic approach uses replicator dynamics to describe the learning *behavior* of various RL algorithms, and often pays less attention to the quantitative performance of the learners (e.g., Tuyls *et al.*, 2006; Panait *et al.*, 2008; Klos *et al.*, 2010).

The aim of this thesis is to combine insights from both approaches in order to arrive at a thorough analysis of the qualitative as well as quantitative aspects of reinforcement learning. This provides a means to link certain behavioral properties of a learning algorithm to its performance. This link is established using various analytical tools to describe the behavior, convergence properties and performance of different learners, both by simulating the actual learning algorithm and by analyzing the corresponding replicator dynamics.

The strength of the evolutionary game theoretic approach is demonstrated by introducing a variation of Q-learning that is based on insights from evolutionary game theory. Recently, it has been shown that the introduction of leniency to the evolutionary model of Q-learning improves convergence to Pareto optimal equilibria in cooperative games (Panait *et al.*, 2008). However, discrepancies have been observed between between the predicted and actual behavior of Q-learning (Kaisers and Tuyls, 2010), and it is argued that these discrepancies carry over to Lenient Q-learning as well. This thesis proposes a practical learning algorithm, Lenient Frequency Adjusted Q-learning, that implements the evolutionary model, thereby resolving the discrepancies. The match between the model and the proposed learning algorithm is argued theoretically and demonstrated empirically.

This thesis further contributes by investigating different state-of-the-art learning algorithms both in homogeneous as well as in heterogeneous environments, thereby being able to compare results from both scenarios. In a homogeneous environment, all agents learn using the same learning algorithm; in a heterogeneous environment multiple different learners interact. Whereas in the traditional, simulation-centered approach the idea of heterogeneous learning is not new (e.g., Bab and Brafman, 2008), the EGT approach so far has mainly considered homogeneous learning. Therefore, this thesis contributes explicitly to the EGT approach by employing replicator dynamics in the analysis of heterogeneous learning.

The above considerations allow to define the following general problem statement:

Problem statement. *How can evolutionary game theory be used as a framework to analyze multi-agent reinforcement learning algorithms in a heterogeneous setting; and how can a learning algorithm be derived that implements a given evolutionary model?*

This problem statement is refined in three research questions, that highlight the main problems indicated above. First of all, the discrepancy between the predicted and actual behavior of Lenient Q-learning needs to be resolved, in order to provide a proper comparison of the learning algorithms that are considered. This leads to the first research question.

Research question 1. *Does the proposed Lenient Frequency Adjusted Q-learning algorithm effectively implement the evolutionary model derived by Panait* et al. *(2008), thereby resolving the discrepancies observed between the actual and predicted behavior of Lenient Q-learning?*

Secondly, the behavior, convergence properties, and performance of different reinforcement learning algorithms need to be analyzed, both in homogeneous and in heterogeneous settings, by combining the traditional algorithmic approach and the evolutionary game theoretic approach. A thorough analysis of multi-agent reinforcement learning using both approaches provides a next step towards a stronger theoretical framework. This results in the second research question.

Research question 2. *To what extent can the evolutionary game theoretic approach facilitate the analysis of multi-agent reinforcement learning algorithms in homogeneous environments, and can these insights be effectively generalized to heterogeneous environments?*

Finally, these insights can lead to a better understanding of how evolutionary game theory can contribute in analyzing the link between behavior and performance of multi-agent reinforcement learning algorithms. This understanding is required in order to establish evolutionary game theory as a theoretical framework for multi-agent reinforcement learning. The final research question is therefore as follows.

Research question 3. *How can the traditional algorithmic approach and the evolutionary game theoretic approach complement each other in order to analyze the link between behavior and performance of multi-agent reinforcement learning algorithms?*

In order to answer these research questions, the theoretical discussion is complemented by an empirical analysis of reinforcement learning applied to 2×2 normal form games.

1.4 Outline

The remainder of this thesis is structured as follows.

Chapter 2 introduces the topic of reinforcement learning, and presents the learning algorithms studied in this thesis. These are Q-learning, Finite Action-set Learning Automata, and Polynomial Weights Regret Minimization. Two recent variations of Q-learning are described: Lenient Q-learning and Frequency Adjusted Q-learning. Furthermore, Lenient Frequency Adjusted Q-learning, a combination of both, is proposed.

Chapter 3 gives an overview of the relevant game theoretic background, including normal form games, Nash equilibria, and Pareto optimality. It then proceeds to describe concepts from evolutionary game theory, most notably evolutionarily stable strategies and the replicator dynamics. Throughout the chapter examples of games are given, of which the Prisoners' Dilemma, the Stag Hunt game, and the Battle of the Sexes are used recurrently in later chapters.

Chapter 4 presents the formal relation between evolutionary game theory - in particular the replicator dynamics - and reinforcement learning. It recapitulates the work of Börgers and Sarin (1997), who first proved that Cross learning converges to the replicator dynamics. Recently established evolutionary models of (Lenient) Q-learning, Learning Automata, and Regret Minimization are presented. Furthermore, the variation Frequency Adjusted Q-learning is described in detail, and it is argued that this variation can be applied to Lenient Q-learning as well.

Chapter 5 outlines the methodology used for the experiments that are conducted. The games that are used to test the learning algorithms are introduced, parameter tuning is described, and the setup of the experiments is presented. Furthermore, the chapter describes the analytical tools that are used to analyze the behavior, convergence properties and performance of the learning algorithms.

Chapter 6 presents the results of the experiments. It confirms that frequency adjustment indeed applies to Lenient Q-learning as well, and presents a rigorous analysis of the influence of the learning step size on the behavior of the learners. A detailed description is given of the behavior, convergence properties and performance of the learning algorithms in homogeneous and heterogeneous environments.

Finally, Chapter 7 concludes this thesis by summarizing and discussing the results, and provides answers to the research questions posed in the previous section.

Chapter 2

Reinforcement learning

Reinforcement learning (RL) is a specific type of interaction-based learning. It differs from both supervised and unsupervised learning. Where a supervised learner receives full feedback on its actions by means of input/output pairs, the reinforcement learner only receives an immediate reward and has to learn from experience whether this reward is good or not. An unsupervised learner on the other hand receives no feedback on the quality of its action at all, which puts the reinforcement learner somewhere in the middle with respect to the amount of feedback that is provided (Sutton and Barto, 1998).

This chapter starts with a basic introduction to the theory of reinforcement learning. Section 2.2 then proceeds to describe some well-know single-agent learning algorithms. Section 2.3 presents the challenges involved in multi-agent reinforcement learning, together an example of a RL algorithm that is specifically suited to the multi-agent setting. Finally, Section 2.4 discusses several additional multi-agent RL algorithms to provide a more broad overview of the field.

2.1 The reinforcement learning problem

A reinforcement learning agent has to learn by trial-and-error interaction with its environment. It has no explicit knowledge on how to achieve its goal, it can only perceive the results of its actions by means of punishment and reward. The agent's goal is to maximize its reward over time by learning what the best action is in each situation. A reinforcement learning algorithm is any learning algorithm that aims to solve this problem. Therefore, reinforcement learning can best be characterized by a set of learning *problems* rather than by a set of learning *algorithms* (Kaelbling *et al.*, 1996; Sutton and Barto, 1998).

More formally, the RL problem consists of a set of environment states, S; a set of the agent's actions $a(s)$ in each state; and a set of rewards, e.g., $r \in [0, 1]$. Each time step the agent perceives the state $s(t) \in S$ of its environment and performs action $a_i(t) \in a(s)$, upon which it receives a reward $r_i(t+1)$ and the environment moves to state $s(t+1)$. Figure 2.1 shows this agent-environment interaction graphically.

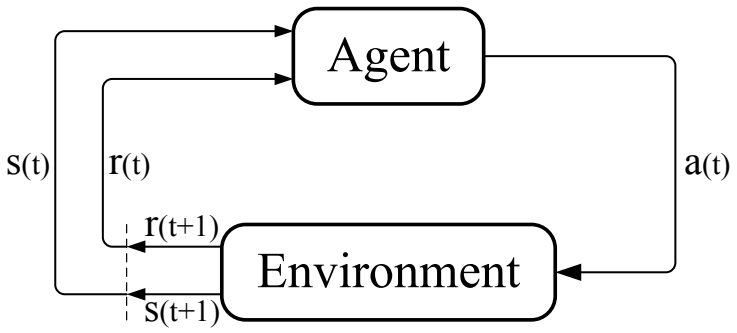


Figure 2.1: The reinforcement learning model of interaction (Sutton and Barto, 1998).

The agent implements a probability distribution, or policy, x that maps states of the environment to probabilities over the agent's actions. This policy is updated each time the agent receives a reward in

such a way that over time it converges to the optimal policy x^*. The way in which the policy is updated is specified by the reinforcement learning algorithm used.

Optimality in reinforcement learning means achieving the highest cumulative reward $R(t) = r(t+1) + r(t+2) + \ldots + r(T)$, where T is the final time step. In settings where the learning problem is unbounded, i.e., where there is no final time step, this poses a problem as $R(t) \to \infty$. To solve this problem a discount factor $\gamma \in [0, 1)$ is introduced to bound the expected future reward. By choosing $\gamma < 1$ the infinite sum of discounted rewards converges and the learning problem is no longer unbounded. This results in the following definition of the cumulative reward (Sutton and Barto, 1998; Tuyls, 2004):

$$R(t) = \sum_{k=1}^{\infty} \gamma^k r(t + k + 1). \tag{2.1}$$

The discount factor determines what an expected future reward is worth to the agent at present: the value of a reward received k time steps ahead is decreased by a factor γ^{k-1}. If $\gamma = 0$ the agent is short-sighted; it only considers the immediate reward $r(t + 1)$. If the reward is close to 1 the agent is far-sighted. By properly choosing the discount factor the agent can be driven toward early rewards and shorter solutions.

2.1.1 Exploration - exploitation dilemma

Choosing which action to take is an important aspect of reinforcement learning. Should the agent exploit actions that proved to be good in the past to ensure a high immediate reward, or should it explore in order to achieve better results in the future, thereby risking a low reward now? Neither of the two is sufficient alone, and the dilemma is to find the right balance. This is known as the *exploration-exploitation dilemma* (Kaelbling *et al.*, 1996; Sutton and Barto, 1998).

Several methods have been devised that control this balance between exploration and exploitation. Among these are greedy strategies, that choose actions with the highest estimated reward, and randomized strategies that with a certain probability also explore a random action (Kaelbling *et al.*, 1996). An example of the latter is the Boltzmann exploration mechanism, which is described in detail in Section 2.2.1. Another solution is to start exploring, and gradually switch to exploitation during the learning process. The underlying idea is that in the beginning exploration is needed in order to estimate the pay-off that results from an action, whereas later on exploitation becomes more important to ensure high cumulative rewards.

2.1.2 Stateless reinforcement learning

The domain of RL problems used in this thesis is limited to that of repeated interactions between agents in a single state environment. A number of agents encounters each other, they independently select and perform an action, and each receives a reward. The agents then update their policy based on this reward, and the next iteration starts. As a result, the environment of this type of interaction is static, i.e., no state transitions occur.

In a static environment each agent only has a single set of actions $a = \{a_1, \ldots, a_n\}$ to choose from. The agent's policy is now simply a probability distribution over its actions, $x = \{x_1, \ldots, x_n\}$. The RL algorithms discussed in the remainder of the chapter are all formulated as stateless algorithms. However, it should be noted that in literature often formulation including state information are given, especially in th case of Q-learning.

2.2 Single-agent reinforcement learning

Single-agent reinforcement learning is already an established research area with a firm theoretical basis (Kaelbling *et al.*, 1996; Sutton and Barto, 1998). The most notable advantage of restricting oneself to only one agent is that the environment is *stationary* and can be treated as Markovian, i.e., the current state provides as much information as the complete history of states so far. This greatly simplifies the learning problem and allows the construction of proofs of convergence to optimal policies for several learning algorithms, e.g., Q-learning (Watkins and Dayan, 1992).

There are two basic types of RL algorithms: *value-based* and *policy-based* learners. Value-based learners estimate an action-value function Q that captures the long-term reward of an action and derive

a policy from that, whereas policy-based learners update their policy directly based on the reward received (Van den Herik *et al.*, 2007; Kalyanakrishnan and Stone, 2009). An example of a value-based learner is Q-learning (see Section 2.2.1); learning automata and regret minimization are examples of policy-based algorithms (Sections 2.2.2 and 2.2.3).

2.2.1 Q-learning

Q-learning is an off-policy reinforcement learning algorithm based on the idea of temporal difference learning (Watkins and Dayan, 1992). Temporal difference learners generally consist of two steps: policy evaluation and policy iteration. The first step estimates a value function, that is then used in the second step to update the policy. Q-learning differs from on-policy TD learners in that it approximates the true action-value function Q^* independent of the policy being followed (Sutton and Barto, 1998).

The standard form of Q-learning uses the action-value update function

$$Q_{a_i}(t+1) \leftarrow Q_{a_i}(t) + \alpha \left[r_i(t+1) + \gamma \max_j Q_{a_j}(t) - Q_{a_i}(t) \right] \tag{2.2}$$

to refine Q at every time step, where a_i is the action taken at time t, α is a step size parameter, and γ the discount factor. Only the value of the selected action is updated; for all other actions a_j, $Q_{a_j}(t+1) \leftarrow Q_{a_j}(t)$. The policy plays no role in this update process; it is only used to determine which action is selected. Instead, the action-value update is based purely on the reward received and the expected value of the next iteration, expressed by the term $\gamma \max_j Q_{a_j}(t)$. After each update of Q, the new optimal policy is derived using an action selection mechanism that converts the action-value function Q to the probability distribution x.

Two often used action selection mechanisms are *ϵ-greedy* and *Boltzmann exploration* (Sutton and Barto, 1998). ϵ-Greedy selects the best action (with the highest Q-value) with probability $(1-\epsilon)$, and with probability ϵ it selects an action at random, independent of the Q-values. The Boltzmann exploration mechanism makes use of a temperature parameter τ that controls the balance between exploration and exploitation. It selects action a_i with probability

$$x_i = \frac{e^{Q_{a_i} \cdot \tau^{-1}}}{\sum_j e^{Q_{a_j} \cdot \tau^{-1}}}. \tag{2.3}$$

A high temperature drives the mechanism towards exploration by leveling the action probabilities, whereas a low temperature promotes exploitation by favoring actions with a high Q-value.

Q-learning is proven to converge to the true action-value function Q^* in a Markovian environment, given that each action is selected (and its action-value is updated) an infinite number of times (Watkins and Dayan, 1992).

2.2.2 Learning Automata

Learning automata differ from Q-learning in that they do not estimate a value function but learn directly in the policy space. Traditionally, these automata assume a finite set of actions, and are therefore called *finite action-set learning automata* (FALA) (Thathachar and Sastry, 2002). Other, more elaborate automata such as *parameterized*, *generalized* and *continuous action-set* learning automata exist but these fall outside the scope of this thesis. For an overview, the interested reader is referred to Thathachar and Sastry (2002).

Learning in a FALA proceeds along the same steps as other reinforcement learning algorithms. At each time step the automaton draws a random action a_i according to its policy x. Based on this action it receives a reward r_i which it then uses to update its policy $x(t) \rightarrow x(t+1)$. The update rule for FALA is

$$x_i(t+1) \leftarrow x_i(t) + \alpha r_i(t+1)(1 - x_i(t)) - \beta(1 - r_i(t+1))x_i(t) \tag{2.4}$$
$$\text{if } a_i \text{ is the action taken at time } t \text{ or}$$
$$x_j(t+1) \leftarrow x_j(t) - \alpha r_i(t+1)x_j(t) + \beta(1 - r_i(t+1)) \left[(k-1)^{-1} - x_j(t) \right] \tag{2.5}$$
$$\text{for all } a_j \neq a_i$$

where $\alpha, \beta \in (0, 1)$ are the reward and penalty parameters respectively, and k is the total number of actions. Various settings for α and β result in different update schemes. Often used schemes are *linear reward-penalty* (L_{R-P}) when $\alpha = \beta$, *linear reward-inaction* (L_{R-I}) when $\beta = 0$ and *linear reward-ϵ-penalty* ($L_{R-\epsilon P}$) when $\alpha >> \beta$ (Tuyls, 2004).

2.2.3 Regret Minimization

Just like learning automata, regret minimization (RM) also learns directly in the policy space (e.g. Jafari *et al.*, 2001; Blum and Mansour, 2007). This method is based on the idea that an agent incurs a certain loss for not playing optimally all the time. This typically results from the uncertainty of the environment; the agent has to explore in order to find a better policy and will therefore play suboptimal in the beginning. In hindsight, the agent might *regret* not having played differently. RM algorithms aim to update the agent's policy in such a way that this regret is minimized.

In general, an agent playing policy x incurs a certain loss L with respect to some alternative policy x'. The higher this loss, the more the agent regrets having played x. The alternative policy can be defined in different ways, leading to different notions of regret. In the case of *external regret*, x' simply chooses the best fixed action in hindsight. Alternatively, in the case of *internal regret*, or *swap regret*, x' changes every occurrence of a given action a_i to an alternative action a_j. Note that these definitions both assume that there exists a best action for each situation, whereas other RL algorithms only assume that there is a best *mix* of actions. The latter might be more plausible given the stochastic nature of many environments.

Another important characteristic of a RM algorithm is the amount of information that is available to the agent. When the agent knows, in hindsight, the value of all its actions at each time step it is said to have *full information*. In this case, it can calculate the loss for each individual action against the best action in hindsight and update their probabilities accordingly. More often, however, the agent might only have *partial information*, i.e., it only knows the reward for its previous action. The agent can now only estimate the loss for this single action based on the history of actions and rewards so far, and update its probability.

The RM algorithm used in this thesis is based on the assumptions of *external regret* and *full information*. This means that at each time step all action probabilities are updated based on the action's loss with respect to the maximum reward in hindsight. Suppose the agent has taken action a_i. It will then receive reward r_i as usual, but it will also get information about all other rewards r_j it could have received had it played differently. The agent can now calculate the loss l for each action based on the maximum reward r^* as $l_i = r^* - r_i$. A step size parameter λ controls the learning rate when updating the weights w that are assigned to each action:

$$w_i(t+1) \leftarrow w_i(t)\left(1 - \lambda l_i(t+1)\right). \tag{2.6}$$

The agent can now derive its new policy by normalizing the weights according to

$$x_i(t) = \frac{w_i(t)}{\sum_j w_j(t)}. \tag{2.7}$$

As a result, the weight of actions that incurred a loss will decrease, and so will their probability of being selected. On the other hand, the weight of the action that turned out to be the best in hindsight will stay the same as its loss $l_i = 0$. This algorithm is known as the Polynomial Weights (PW) algorithm (Blum and Mansour, 2007; Klos *et al.*, 2010).

The limitations imposed by the assumption of full information might seem severe. Often, the agent will not have access to all information. However, it has been shown that a full information external regret algorithm can be converted to a partial information algorithm that still achieves a low regret (Blum and Mansour, 2007).

2.3 Multi-agent reinforcement learning

Learning in a multi-agent setting is inherently more complex than in the single-agent case described in the previous section. The learning problem can either become *competitive*, *cooperative*, or a combination of

both. In a competitive environment the one agent's success might mean the other agent's failure, i.e., the agents have different interests or goals. A cooperative environment on the other hand means that agents have to work together to achieve some common goal. This makes it more difficult to specify the exact goal of the learning process, since mere maximization of individual rewards might not lead to the best overall solution. The exploration-exploitation dilemma also becomes more involved, since exploration is not only necessary in the beginning to find the optimal action but remains important for a longer period of time to account for changes in the reward function due to other agents' changing policy. The fact that the reward function may depend on the actions of other agents shows the most important characteristic of multi-agent reinforcement learning (MARL): the environment is *non-stationary* and as a result each agent is essentially pursuing a moving target (Busoniu *et al.*, 2008).

This section describes how single-agent RL algorithms can be used in a multi-agent setting, by either ignoring the other agents' presence or by modeling it explicitly. Furthermore, an extension of Q-learning that is especially suited to multi-agent coordination is introduced, as well as a recent modification resulting from research on the evolutionary dynamics of Q-learning.

2.3.1 Single-agent learning in a multi-agent setting

Single-agent RL algorithms can also be applied to multi-agent learning, but not without consequences. Two extreme approaches can be distinguished that deal with the problem of MARL: the *joint action space* approach on the one hand and *independent learning* on the other (Tuyls and Nowé, 2005). Both have their strengths and weaknesses.

In the joint action space approach the influence of one agent on all others is modeled explicitly. As the name indicates, learning happens in the joint space of all agent's action sets. As a result, the state transition function is a direct mapping from a state and an action from each agent to a probability distribution over the set of states S. This way, the current state still contains as much information as the full history of states and the Markov property still holds. However, the joint actions space approach ignores some of the basic principles of multi-agent systems: the need for complete information requires free and instant communication, which is in general not feasible, and leaves no room for distributed control.

The opposite approach is to use independent reinforcement learners. These agents have no way of communicating with each other, and treat the problem as if no other agents were present. As a result, all single-agent RL algorithms discussed in Section 2.2 can also be used by in a multi-agent setting. There is however a major drawback. Since the feedback of the environment is dependent on the actions taken by *all* agents, for an independent learner the environment is no longer stationary and the Markov property does not hold anymore. This means that general proofs of convergence for the single-agent case can not be applied to the multi-agent setting. Despite this lack of a strong theoretical framework, independent learning has been shown empirically to produce good results (Busoniu *et al.*, 2008; Tuyls and Nowé, 2005). Therefore, this thesis focuses on independent learners only.

2.3.2 Lenient Q-learning

Lenient Q-learning is a learning algorithm specifically tailored to cooperative multi-agent environments. When multiple independent agents learn together in such an environment, it can often happen that they converge to suboptimal solutions whereas a single agent might have no difficulty at all in finding the optimum. One of the reasons is that the environment is unpredictable, since actions taken by other agents influence both the rewards received and the state transitions that occur. For example, consider two agents Ag_1 and Ag_2 learning to play soccer together, and assume they have no prior knowledge of each other's skill. It might very well happen that even a perfect forward pass by Ag_1 is not rewarded with a goal since Ag_2 still lacks the skill or experience to deal with it properly. Using traditional RL algorithms, Ag_1 might decide that this is not a good pass (since there was no reward) and not play it again. However, in doing so it completely ignores that fact that Ag_2 is also still learning and might in the future be able to score from that position. In this case, a possibly very highly rewarded combination of actions might be lost because of initial errors made in the beginning. Had Ag_1 shown some *lenience* towards Ag_2 it could have forgiven the early mistake, leading to more optimal behavior in the long run (Panait *et al.*, 2008).

Leniency towards others can be achieved by having the agent ignore some low rewards and only consider the highest of several samples. For example, the agent might always update its policy when an action yields a higher reward than expected. When the reward is lower, it chooses probabilistically: in the beginning it should show more lenience towards its teammates and ignore more low rewards, whereas in the end it might be more critical and always update its policy (Panait, Sullivan, and Luke, 2006). A more simple approach is to have the agent collect κ rewards for a single action before it updates the value of this action according to the *highest* of those κ rewards (Panait *et al.*, 2008). This results in a fixed degree of leniency, expressed by the value of κ.

It has been shown that leniency can greatly improve the accuracy of an agent's projection of the search space in the beginning of the learning process (Panait *et al.*, 2006). It thereby overcomes the problem that initial errors by other agents might lead to suboptimal solutions in the long run. This thesis considers a lenient version of the Q-learning algorithm described in Section 2.2.1, therefore referred to as Lenient Q-learning (LQ).

2.3.3 (Lenient) Frequency Adjusted Q-learning

Recently, evolutionary dynamical models of Q-learning and Lenient Q-learning have been developed with the intention to provide a more intuitive way to understand the behavior of reinforcement learning (Tuyls *et al.*, 2003b; Panait *et al.*, 2008). However, it has been shown that the behavior of Q-learning deviates from the predictions of its evolutionary model, and that the behavior displayed by the evolutionary model is in fact more desirable than the actual behavior of Q-learning. The source of this discrepancy proved to be the fact that the Q-value update function is unbalanced: actions that are selected more often, are also updated more often. The evolutionary model, on the other hand, assumes that all Q-values are updated infinitely often, which is not the case in practice. This led to a modification of Q-learning, in which an action's Q-value update is inversely proportional to the probability with which that action is selected. This modified Q-learning algorithm is called Frequency Adjusted Q-learning (FAQ) (Kaisers and Tuyls, 2010).

Logically, it can be assumed that the same reasoning applies to Lenient Q-learning, since its evolutionary model is derived in a way similar to the model of Q-learning. It is shown in this thesis that the behavior of Lenient Q-learning indeed deviates from its evolutionary predication. Furthermore, a lenient version of the FAQ algorithm, called Lenient Frequency Adjusted Q-learning (LFAQ), is proposed in this thesis that adheres to the predictions of the evolutionary model of Lenient Q-learning in the same way that FAQ adheres to the model of Q-learning. The mathematical details of both frequency adjusted algorithms are presented in Section 4.2.

2.4 Related work

As mentioned earlier, the non-stationary nature of multi-agent environment violates the Markov property, thereby rendering proofs of convergence for traditional RL algorithms inapplicable to the multi-agent setting. As a result, several authors came up with modifications of standard RL techniques, such as the LQ algorithm described in Section 2.3.2, or even new algorithms that are specifically designed for multi-agent learning. This section provides an overview of some of these techniques. Note that these algorithms are described here only to present a broader view on the field of MARL; they are not included in the experiments performed in this thesis.

Kapetanakis and Kudenko (2002) present the Frequency Maximum Q (FMQ) technique, an extension of Q-learning that modifies the agent's action selection strategy. FMQ serves as a heuristic that is applied to the Boltzmann policy update scheme (Equation 2.3). Instead of using the Q values directly to calculate the policy, the FMQ heuristic uses expected values that depend also on the maximum reward r_i^* for an action received so far, the frequency f with which this maximum reward is encountered, and a weight c that controls the importance of the FMQ heuristic:

$$EV_{a_i} = Q_{a_i} + c * f(r_i^*) * r_i^*. \tag{2.8}$$

The authors show that agents using the FMQ heuristic are better able to coordinate their actions in games where miscoordination is severely punished, such as the penalty game and the climbing game.

Other authors have taken a different approach by designing new algorithms that are not based on traditional RL techniques. An example is the work by Singh, Kearns, and Mansour (2000), in which the authors use a *gradient ascent* algorithm (IGA, for Infinitesimal Gradient Ascent) to tackle the problem of multi-agent learning. They are able to provide a proof of convergence for a restricted class of multi-agent games, which is a rather interesting result since it is one of the first, albeit weak, convergence proofs for a multi-agent learning algorithm (Bowling and Veloso, 2002).

This result is built upon by Bowling and Veloso (2002) who introduce the idea of a *variable learning rate*. The idea is that an agent should adapt quickly when it receives low rewards, whereas it should be more cautious when doing better than expected in order to keep doing well. This Win or Learn Fast (WoLF) algorithm greatly depends on what constitutes 'winning'. The authors define that an agent is winning when it prefers its current policy over some known Nash equilibrium policy, otherwise it is losing. They further provide a convergence proof for a combination of WoLF with the gradient ascent algorithm (IGA-WoLF) that is strictly stronger than the initial proof given by Singh *et al.* (2000).

A generalization of the IGA algorithm is described by Zinkevich (2003). Where IGA only applies to two-player games with two actions, the generalized IGA (GIGA) applies to games with an arbitrary number of actions. However, both GIGA and its extension GIGA-WoLF have rather severe assumptions with respect to the amount of information that is available to the agents. Both algorithms assume knowledge of the underlying game, specifically its Nash equilibria. In most real-world situations this kind of knowledge might not be available, which limits the practical applicability of these algorithms.

Abdallah and Lesser (2008) propose a Weighted Policy Learner (WPL) algorithm that follows the same line of reasoning as the GIGA-WoLF algorithm but with less stringent information requirements. Unlike (G)IGA-WoLF, WPL does not need any knowledge about the underlying game, such as the Nash equilibria, a priori and is therefore better suited for practical implementation. The authors show that WPL, despite its lack of prior information, performs on similar level as IGA-WoLF in several two-player games. Moreover, when a larger number of agents are involved, WPL performs better than both GIGA and GIGA-WoLF.

Chapter 3

Evolutionary game theory

Game Theory (GT) is an economical theory that studies multi-player decision problems (Gibbons, 1992). These problems typically arise when two or more players have to make a decision, thereby taking into account not only their own but also the other players' actions. Classical game theory assumes that full information is available to the player, which together with the assumption of hyper-rationality does not reflect the dynamical nature of most real world environments (Gintis, 2009). Evolutionary Game Theory (EGT) was developed to overcome this limitation, by adopting the idea of evolution from biology to describe how agents can learn to optimize their behavior without having complete information (Maynard Smith and Price, 1973; Tuyls and Nowé, 2005). The theory provides a solid basis to study the decision making process of boundedly rational players in an uncertain environment.

This chapter presents several basic concept from classical game theory, such as the Nash Equilibrium and Pareto optimality. These concepts are then built upon to describe the basics of evolutionary game theory. Most notably, the Replicator Dynamics are introduced, and their relation to Evolutionarily Stable Strategies and the Nash Equilibrium is described. The theory of the replicator dynamics forms the bridge between evolutionary game theory and multi-agent reinforcement learning. This connection is the topic of Chapter 4.

3.1 Game theoretic background

Game theory models the interaction between players as a game, in which each player has a set of actions to choose from. All players have to select an action simultaneously, upon which they receive a payoff that depends on the combination of actions played. The goal for each player is to come up with a strategy that maximizes its payoff in the game. It is assumed that the players are rational, in the sense that each player tries to maximize its own payoff irrespective of the payoffs of the others (Gibbons, 1992).

Note that although the concepts used to describe game theory are very similar to those in multi-agent learning, the terminology differs slightly. Where multi-agent learning deals with agents, environments and policies, game theory involves players, games and strategies.

3.1.1 Games and strategies

In its most simple form, a game involves two players with two actions each, also denoted as a 2×2 game. It is this type of games that are studied in this thesis, however most concepts and ideas apply also to the more general case of n-player games with arbitrary numbers of actions for each player. In the normal-form representation of a game, each player selects an action simultaneously, and the combination of actions played determines the payoff to each player (Gibbons, 1992).

Definition 1. *The* ***normal-form representation*** *of a 2-player game specifies the players' strategy spaces S_1, S_2 and their payoff functions $P_1, P_2 : S_1 \times S_2 \rightarrow \Re$. The game is denoted by $G = \{S_1, S_2; P_1, P_2\}$.*

A player's strategy space S consist of all possible probability distributions over the player's actions. In the case of 2×2 games, this entails $S = \{s = (x_1, 1 - x_1) : x_1 \in [0, 1]\}$.

A well known example of such a game is the Prisoners' Dilemma (Gibbons, 1992):

> "Two suspects are arrested and charged with a crime. The police lack sufficient evidence to convict the suspects, unless at least one confesses. The police holds the suspects in separate cells and explain the consequences that will follow from the actions they could take. If neither confesses then both will be convicted for a minor offense and sentenced to one month in jail. If both confess then both will be sentenced to jail for six months. Finally, if one confesses but the other does not, then the confessor will be released immediately but the other will be sentenced to nine months in jail - six for the crime and further three for obstructing justice."

In this game, the players each have the choice to cooperate (with each other, i.e., stay silent) or defect (confess). For notational convenience, the negative payoffs (jail time) can be converted into positive payoffs (reduced sentence with respect to the maximum penalty). The numbers can be chosen arbitrarily, as long as they accurately represent the game's intention. For example, Tuyls (2004) uses the values 5, 3, 1 and 0 to represent 0, 1, 6, and 9 months jail time, respectively. These payoffs can be replaced by any combination T, R, P, and S (for temptation, reward, punishment, and sucker) as long as the relation $T > R > P > S$ holds. The options and corresponding payoffs to each player can be summarized in matrix form, as in Figure 3.1. Matrix A has to be seen from the row perspective, and B from the column perspective. For example, matrix A shows the payoff for the row player when playing either of the actions Cooperate (C) or Defect (D) in the corresponding rows, where the payoffs then depend on the action of the other player (and similarly so for matrix B).

$$A = \begin{matrix} C \\ D \end{matrix} \begin{pmatrix} 3 & 0 \\ 5 & 1 \end{pmatrix} \qquad B = \begin{matrix} \\ \end{matrix} \begin{matrix} C & D \\ \end{matrix} \begin{pmatrix} 3 & 5 \\ 0 & 1 \end{pmatrix}$$

Figure 3.1: Payoff matrices for the Prisoners' Dilemma, with the actions Cooperate (C) and Defect (D). Matrix A defines the payoff for the row player, B for the column player.

These payoff matrices can conveniently be combined in the bi-matrix (A, B) as shown in Figure 3.2. A bi-matrix can, just like a normal matrix, contain any number of rows and columns; "bi" just reflects the fact that each cell contains two numbers: the payoffs for both players (Gibbons, 1992). Suppose the row player plays action i and the column player plays j, then the bi-matrix (A, B) gives the payoffs A_{ij} to the row player and B_{ij} to the column player.

$$\begin{matrix} \\ C \\ D \end{matrix} \begin{matrix} C & D \\ \end{matrix} \begin{pmatrix} 3,3 & 0,5 \\ 5,0 & 1,1 \end{pmatrix}$$

Figure 3.2: Bi-matrix representation of the Prisoners' Dilemma, the first value in each cell represents the payoff for the row player, the second value for the column player.

A strategy $s_i \in S$ for player i in this game is now simply the choice between action C and D or, in general, a probability distribution over both actions as defined above. The former is called a *pure* strategy, the latter is a *mixed* strategy. Some strategies might be preferred over others, based on the expected reward they entail. For example, in the Prisoners' Dilemma, a rational player will always prefer the action Defect over Cooperate, since no matter what its opponent does, it will always receive a higher reward when playing Defect. Strategy D is said to *strictly dominate* strategy C, defined as follows (Gibbons, 1992):

Definition 2. *In the 2-player normal-form game $G = \{S_1, S_2; P_1, P_2\}$, let s_i and s_i' be feasible strategies for player i. Strategy s_i* ***strictly dominates*** *s_i' if, for every feasible strategy of the other player, i's payoff from playing s_i is strictly higher than its payoff from s_i'.*

A rational player will never play a strictly dominated strategy, since there is no believe it can hold about the other player's strategy such that this strategy would be optimal. As a result, in the *one-shot* Prisoners' Dilemma the outcome (D,D) will be selected by rational players, even though (C,C) would yield a higher payoff for both players.

3.1.2 Nash equilibrium

Related to the concept of strictly dominating strategies is the Nash equilibrium (NE). A set of strategies for all players forms a Nash equilibrium if no single player can do better by playing a different strategy (Gibbons, 1992).

Definition 3. *In the 2-player normal-form game* $G = \{S_1, S_2; P_1, P_2\}$*, the strategy profile* $s = (s_1, s_2) \in S_1 \times S_2$ *is a* ***Nash equilibrium*** *if* s_1 *is the best response of player 1 to the strategy* s_2 *played by player 2, and vice versa:* $P_1(s_1, s_2) \geq P_1(s'_1, s_2)\ \forall s'_1 \in S_1$ *and* $P_2(s_1, s_2) \geq P_2(s_1, s'_2)\ \forall s'_2 \in S_2$*.*

For example, in the one-shot Prisoners' Dilemma the outcome (D,D) is the only Nash equilibrium of the game, even though both players would be better of playing (C,C). The reason is that in the latter case, each player can increase its payoff by deviating and playing D. A Nash equilibrium does not necessarily involve pure strategies, mixed Nash equilibria are also possible. John Nash (1950) proved that any finite game (with a finite number of players and actions) has a Nash equilibrium.

A game can also have multiple Nash equilibria, which is illustrated by the Battle of the Sexes game in Figure 3.3. In this game, a couple has to decide independently what they want to do in the evening. They prefer to do something together, but the male (column player) would rather visit a fighting match, whereas the female (row player) would rather visit the opera, as indicated by the payoff matrix. This game has two pure Nash equilibria, (O,O) and (F,F), and a mixed Nash equilibrium where each plays their preferred action with probability $\frac{1}{3}$.

$$\begin{array}{cc} & \begin{array}{cc} O & F \end{array} \\ \begin{array}{c} O \\ F \end{array} & \left(\begin{array}{cc} 2,1 & 0,0 \\ 0,0 & 1,2 \end{array} \right) \end{array}$$

Figure 3.3: Payoff matrix for the Battle of the Sexes, with the actions Opera (O) and Fight (F).

3.1.3 Pareto optimality

The example of the Prisoners' Dilemma in Figure 3.2 shows that a Nash equilibrium is not necessarily the best outcome for all players, since (C,C) would yield a higher payoff to both players than the Nash equilibrium (D,D). Similarly, in a game with multiple Nash equilibria, one might be preferred over the others. The concept of Pareto optimality is used to capture this idea: an outcome is Pareto optimal if no player can improve its payoff without reducing the payoff to any other player (Gintis, 2009; Tuyls and Nowé, 2005).

Definition 4. *In the 2-player normal-form game* $G = \{S_1, S_2; P_1, P_2\}$*, the strategy profile* $s = (s_1, s_2) \in S_1 \times S_2$ ***Pareto dominates*** *another strategy profile* $s' \neq s$ *if, for all* i*,* $P_i(s) \geq P_i(s')$ *and there exists a* j *such that* $P_j(s) > P_j(s')$*.*

An outcome is Pareto optimal if it is not Pareto dominated by any other outcome. In the Prisoners' Dilemma, the outcome (C,C) is Pareto optimal and it Pareto dominates (D,D).

The Stag Hunt game (Skyrms, 2001) provides an example of a game with multiple Nash equilibria, of which one Pareto dominates the others. First described by the 18^{th} century French philosopher Jean-Jacques Rousseau, the Stag Hunt involves two individuals that go out on a hunt. Each player must choose to hunt either a stag or a hare, without knowledge of the other player's choice. Hunting for a hare is relatively easy, and the chance of getting a hare for one player is independent of what the other player does. Capturing a stag is much more difficult, however, and can only be achieved if both players cooperate. Furthermore, a stag is more valuable than a hare as it provides a bigger meal. Possible payoffs for this game are given in Figure 3.4.

$$\begin{array}{cc} & \begin{array}{cc} S & H \end{array} \\ \begin{array}{c} S \\ H \end{array} & \left(\begin{array}{cc} 4,4 & 1,3 \\ 3,1 & 3,3 \end{array} \right) \end{array}$$

Figure 3.4: Payoff matrix for the Stag Hunt game, with the actions Stag (S) and Hare (H).

The Stag Hunt game has two pure Nash equilibria, (S,S) and (H,H), and one mixed equilibrium where each player chooses to hunt for stag with probability $\frac{2}{3}$. However, only the outcome (S,S) is Pareto optimal in this case. Interesting to note is that the equilibrium (H,H) provides a safe choice for both players, since their payoff is independent of deviations by the other player. For that reason, the Nash equilibrium (H,H) is therefore also called *risk dominant*, whereas (S,S) is called *payoff dominant* (Harsanyi and Selten, 1988).

3.2 Evolutionary game theory

Evolutionary Game Theory (EGT) originated in 1973 with the work of Maynard Smith and Price, who applied game theory to biology (Maynard Smith and Price, 1973). At the time this idea seemed strange, as game theory had always assumed hyper rationality and animals can hardly be said to fit this assumption. To make his idea work, Maynard Smith made three important shifts from traditional game theory with respect to the concepts strategy, equilibrium and player interaction (Gintis, 2009).

Strategy: instead of a player having a strategy set from which to choose, in EGT a *population of players* has a strategy set (variations in genotype) of which individuals inherit one or another (possibly mutated) variant which they then play in their strategic interactions.

Equilibrium: in EGT the Nash equilibrium is replaced by the concept of an *evolutionarily stable strategy* (ESS). A strategy is said to be evolutionarily stable if a population playing that strategy can not be invaded by a small group of mutants.

Player interaction: EGT replaces the one-shot interaction of classical game theory with a *repeated random pairing of players* who play their inherited strategy. The corresponding payoffs to both players determine their fitness, i.e., their reproductive success.

Important to note is that according to the above definitions, a population playing a mixed strategy can be interpreted either as monomorphic, where each individual plays that mixed strategy, or polymorphic, with a fraction of the population playing each of the underlying pure strategies (Gintis, 2009). Although these two interpretations are often interchangeable, the polymorphic representation is preferred in this thesis as it better captures the one-shot nature of the games studied.

3.2.1 Evolutionarily stable strategies

Suppose a population of individuals is playing a certain strategy, and this population is invaded by a group of mutants playing a different strategy. Initially the mutants form only a small fraction of the whole population. If the reproductive success of the mutant strategy is smaller than that of the original strategy, the original strategy will survive and the mutant will disappear. The original strategy is then said to be evolutionarily stable with respect to the invading mutant strategy. If a strategy is evolutionarily stable against all possible mutant strategies, it is said to be an Evolutionarily Stable Strategy (ESS) (Gintis, 2009).

More formally, suppose a large population of players, i, plays the (mixed) strategy s_i. This population is invaded by a small number of agents playing a different strategy s_i'. The invaders initially have a population share of $\epsilon \in (0,1)$. When an individual is playing against a randomly selected opponent, its probability of playing against a mutant is ϵ, and against a non-mutant $1-\epsilon$. This leads to an expected payoff of $P(s_i, (1-\epsilon)s_i + \epsilon s_i')$ if the individual is a non-mutant, and $P(s_i', (1-\epsilon)s_i + \epsilon s_i')$ if it is a mutant. An ESS is then defined as follows:

Definition 5. *A strategy s_i is an **evolutionarily stable strategy** if, for all $s_i' \neq s_i$ and sufficiently small $\epsilon > 0$, $P(s_i, (1-\epsilon)s_i + \epsilon s_i') > P(s_i', (1-\epsilon)s_i + \epsilon s_i')$.*

As an example, it can be proven that the strategy Defect in the Prisoners' Dilemma is an ESS. Assume that in a population of Defectors, a small fraction $\epsilon \in (0,1)$ mutates and becomes Cooperator. The expected payoff for a Defector is now $5\epsilon + 1(1-\epsilon) = 4\epsilon + 1$, and for a Cooperator $3\epsilon + 0(1-\epsilon) = 3\epsilon$. Now, for all ϵ, $4\epsilon + 1 > 3\epsilon$ and therefore Defect is an ESS.

Although in this example the Nash equilibrium and the ESS of the game are the same, the concept of ESS is actually a refinement of the NE. This is clear when looking at an alternative definition of an ESS (Maynard Smith and Price, 1973):

Definition 6. *A strategy s_i is an **evolutionarily stable strategy** if and only if, for any mutant $s'_i \neq s_i$, the following conditions hold:*

1. *$P(s_i, s_i) \geq P(s'_i, s_i)$, and*
2. *if $P(s_i, s_i) = P(s'_i, s_i)$, then $P(s_i, s'_i) > P(s'_i, s'_i)$.*

The first property states that an ESS is a NE, and the second property is the refinement stating that if the invading strategy does as well against the original strategy as the original strategy does against itself, then the original strategy must do better against the invader than the invader does against itself. For example, in the game shown in Figure 3.5, the strategy profile (B,B) is a NE, but is is not an ESS since $P(B, B) = P(A, B)$ and $P(B, A) < P(A, A)$.

$$\begin{array}{cc} & \begin{array}{cc} A & B \end{array} \\ \begin{array}{c} A \\ B \end{array} & \left(\begin{array}{cc} 1,1 & 0,0 \\ 0,0 & 0,0 \end{array} \right) \end{array}$$

Figure 3.5: Payoff matrix of a simple game example.

It is important to note that, where a Nash equilibrium is defined on a strategy profile $s = (s_1, \ldots, s_n)$, an ESS is defined on a single strategy s_i. The concept of ESS therefore only applies to symmetric games, i.e., games where the payoff only depends on the strategies played, not on who is playing them. That is, a symmetric 2×2 game conforms the payoff matrix depicted in Figure 3.6.

$$\begin{array}{cc} & \begin{array}{cc} E & F \end{array} \\ \begin{array}{c} E \\ F \end{array} & \left(\begin{array}{cc} a,a & b,c \\ c,b & d,d \end{array} \right) \end{array}$$

Figure 3.6: General payoff matrix of a 2×2 symmetric game.

Both the Prisoners' Dilemma and the Stag Hunt game are symmetric, but the Battle of the Sexes is not. However, it can be converted to a symmetric game by incorporating the type of player (in this case male or female) in the actions, and assigning both types at the start at random to the players. A pure action could then be defined as "xy", meaning "if I am male, I play x, and if I am female, I play y". The symmetric version of Battle of the Sexes then has four pure strategies: OO, OF, FO, and FF (Gintis, 2009).

3.2.2 Replicator dynamics

In order to study the behavior of a population playing an evolutionary game, it is convenient to represent this game as a dynamical system. At the basis of this evolutionary system are differential equations that govern the rate of change of subpopulations playing a particular strategy. As mentioned above, it is assumed here that individuals play only pure strategies, mixed strategies result when fractions of the population play different pure strategies.

At the hart of any evolutionary process are *selection* and *mutation*. Selection is the process in which fit individuals are selected for a next generation, whereas unfit individuals are discarded. Often this process is represented by reproduction, where fit individuals have a higher chance of reproduction than the less fit ones. Individuals that can reproduce by making (approximately) accurate copies of themselves are called *replicators* (Gintis, 2009). A population then consists of a set of replicators that interact according to some pattern. The process of change over time in the frequency distribution of the replicators is the *evolutionary dynamic* of the population.

In the case of an evolutionary game, the replicators are the different pure strategies that are available, and their reproductive success is defined by their payoff when playing against other pure strategies. At each time step, a game is played by random pairing of two individuals in the population. Their respective payoffs then determine their rate of replication. Note that this definition of an evolutionary game assumes that the game is symmetric, as defined above (see Figure 3.6). In a two player game, this means that payoff matrices A and B are related as $A = B^T$, and therefore the payoffs can be represented by only

matrix A for both players. This leads to the single-population dynamics as described below. The more general case of multi-population dynamics is presented subsequently.

Single-population dynamics

Let $p(t)$ be the total size of a population at time t, with $p(t) = \sum_i p_i(t)$ where p_i is the number of individuals in the population playing strategy s_i. The population state x is defined as $(x_1, \ldots, x_n)$, where x_i is the frequency of individuals playing strategy s_i. This implies that $x_i(t) = \frac{p_i(t)}{p(t)}$, and therefore $p_i(t) = p(t)x_i(t)$.

Now assume that the payoff for playing a certain strategy determines the rate of reproduction, and that parents only live for one time step. The total number of individuals playing strategy s_i at time $t+1$ can then be computed by

$$p_i(t+1) = p_i(t)P(s_i, x) \tag{3.1}$$

where $P(s_i, x)$ is the expected payoff for playing strategy s_i in a population with state x. The frequency of individuals playing s_i at time $t+1$ can then by calculated as

$$\begin{aligned} x_i(t+1) &= \frac{p_i(t+1)}{p(t+1)} \\ &= \frac{x_i(t)P(s_i, x)}{\sum_j x_j(t)P(s_j, x)} \end{aligned} \tag{3.2}$$

A full derivation can be found in Tuyls (2004). The rate of change can now be calculating using the difference $\Delta x_i(t) = x_i(t+1) - x_i(t)$, and substituting 3.2:

$$\begin{aligned} \Delta x_i(t) &= \frac{x_i(t)P(s_i, x)}{\sum_j x_j(t)P(s_j, x)} - x_i(t) \\ &= \frac{x_i(t)(P(s_i, x) - \sum_j x_j(t)P(s_j, x))}{\sum_j x_j(t)P(s_j, x)} \\ \frac{\Delta x_i(t)}{x_i(t)} &= \frac{P(s_i, x) - \sum_j x_j(t)P(s_j, x)}{\sum_j x_j(t)P(s_j, x)} \end{aligned} \tag{3.3}$$

This system represented in Equation 3.3 is known as the discrete time Replicator Dynamics (RD) (Tuyls, 2004). The sum $\sum_j x_j P(s_j, x)$ is the average payoff for the whole population, which for convenience will be abbreviated as $P(x, x)$. Equation 3.3 expresses that the rate of change in the frequency of a strategy s_i is equal to the difference between the expected payoff for playing that strategy, $P(s_i, x)$, and the average payoff over all strategies, $\sum_j x_j(t)P(s_j, x)$. If the strategy does better than average its frequency increases, if it does worse it decreases.

The continuous time replicator dynamics can be found by considering time steps of length δ, and calculating the limit as $\delta \to 0$, which yields

$$\frac{dx_i}{dt} = \frac{x_i(P(s_i, x) - P(x, x))}{P(x, x)} \tag{3.4}$$

The denominator has no influence on the qualitative behavior of the system, as the factor $\frac{1}{P(x,x)}$ is the same for all x_i. It can therefore be dropped from Equation 3.4, leading to the general form of the replicator dynamic (Gintis, 2009; Tuyls, 2004):

$$\frac{dx_i}{dt} = x_i(P(s_i, x) - P(x, x)) \tag{3.5}$$

In a symmetric 2-player game with payoff matrix A, the expected payoff of strategy s_i against a population with state $x = (x_1, \ldots, x_n)$ can be written as

$$P(s_i, x) = \sum_j A_{ij}x_j = (Ax)_i \tag{3.6}$$

and the average payoff of the whole population is

$$P(x,x) = \sum_i x_i \sum_j A_{ij} x_j = x^T A x \tag{3.7}$$

The replicator dynamic of Equation 3.5 can then be written as

$$\frac{dx_i}{dt} = x_i[(Ax)_i - x^T A x)] \tag{3.8}$$

Remember that Equation 3.8 is only valid for symmetric games, where $A = B^T$. The Prisoners' Dilemma and the Stag Hunt game are examples of such symmetric games. The next section describes the replicator dynamic for asymmetric games.

Multi-population dynamics

In asymmetric games such as the Battle of the Sexes, the players can have different action sets and payoff matrices, and therefore $A \neq B^T$. The game is then said to be played by individuals of different populations, hence the term *multi-population* dynamics. For simplicity, only the 2-player case is described here. The two populations each have their own state vector, $x = (x_1, \ldots, x_m)$ for the population of player 1, and $y = (y_1, \ldots, y_n)$ for the population of player 2. The respective $m \times n$ payoff matrices are denoted A and B.

The system of differential equations for player 1 can now be written as

$$\frac{dx_i}{dt} = x_i[\sum_{k=1}^{n} A_{ik} y_k - \sum_{r=1}^{m} x_r (\sum_{k=1}^{n} A_{kr} y_k)] \tag{3.9}$$

where $\sum_{k=1}^{n} A_{ik} y_k$ is the average payoff for player 1 when playing strategy s_i, and $\sum_{r=1}^{m} x_r (\sum_{k=1}^{n} A_{kr} y_k)$ is the average payoff over all $s_r \in S_1$ (Tuyls, 2004). Similarly, the system of equations for player 2 is

$$\frac{dy_i}{dt} = y_i[\sum_{k=1}^{m} B_{ki} x_k - \sum_{r=1}^{n} y_r (\sum_{k=1}^{m} B_{kr} x_k)] \tag{3.10}$$

This system can be rewritten similar as in Equations 3.6 and 3.7, resulting in the following two-population replicator dynamics:

$$\frac{dx_i}{dt} = x_i[(Ay)_i - x^T A y] \tag{3.11}$$

$$\frac{dy_i}{dt} = y_i[(Bx)_i - y^T B x] \tag{3.12}$$

Note that the two-population replicator dynamics reduce to single-population dynamics if $A = B^T$, i.e., when the game is symmetric, and assuming that both populations start in the same state ($x = y$).

3.2.3 Relating NE, ESS and the RD

Now that the dynamics of an evolutionary game are known, it is interesting to relate the properties of these dynamics to the (evolutionary) game theoretic properties defined earlier, such as the Nash equilibrium and ESS. As Section 3.2.1 already indicated, the ESS concept is a refinement of Nash equilibria: $ESS \subseteq NE$. This means that every ESS is also a NE, but the inverse is not true. The question is how the two concepts of NE and ESS relate to the replicator dynamic derived in Section 3.2.2.

Central in the analysis of a dynamical system are its *fixed points*, or equilibria. A fixed point is defined as a state x in which the system becomes stable.

Definition 7. *The population state x^* is a **fixed point** of the replicator dynamic if, for all i, $\frac{dx_i}{dt} = 0$.*

In order to describe the behavior of a system around a fixed point, some definitions are needed. Suppose a system is at a point $x(0)$ at time t_0. Then, the *trajectory* through $x(0)$ is the combination of

the collection of points through which the system passes as $t \to \infty$, and the collection of points through which the system passes as $t \to -\infty$.

For a point x, $B_r(x)$ is a *ball of radius* r around x, defined as the set of points y whose distance from x is less than r, with $r > 0$. A trajectory $x(t)$ *approaches* the fixed point x^* if $x(t) \to x^*$ as $t \to \infty$, and it ϵ-*escapes* x^* if there is some t_0 such that $x(t) \notin B_\epsilon(x^*)$ for all $t > t_0$. The set of points $x(0)$ such that the trajectory through $x(0)$ approaches x^* is called the *basin of attraction* of x^*.

It is now possible to say something about the stability of a fixed point x^*.

Definition 8. *A fixed point x^* is **asymptotically stable** if there is some $\epsilon > 0$ such that for all $x(0) \in B_\epsilon(x^*)$, the trajectory through $x(0)$ approaches x^*.*

If x^* is not asymptotically stable, but there is a ball $B_\delta(x^*)$ such that for all $x(0) \in B_\delta(x^*)$ the trajectory through $x(0)$ does not ϵ-escape x^*, x^* is called *neutrally stable*. If x^* is neither asymptotically nor neutrally stable, it is called *unstable*. Finally, if all points where the differential equation is defined lie in the basin of attraction of x^*, the fixed point is called *globally stable*. For a more elaborate introduction to these properties of a dynamical system, the interested reader is referred to Gintis (2009); for an extensive description of dynamical systems in general see Hirsch and Smale (1974).

With these definitions in in mind it is now possible to formally define the relation between Nash equilibria, ESS and the RD. Three important properties of this relation are (Gintis, 2009):

Property 1. *Every Nash equilibrium is a fixed point of the replicator dynamics.*

Property 2. *If a fixed point of the replicator dynamic is asymptotically stable, then it is a Nash equilibrium.*

Property 3. *If strategy s is an ESS, then the population state $x = s$ is an asymptotically stable fixed point of the replicator dynamic.*

These properties make it possible to make some qualitative statements about the equilibria of a game by analyzing the corresponding replicator dynamics. Some examples are given in the next section.

3.2.4 Examples

The replicator equations (3.11 and 3.12) allow to plot the directional field corresponding to the dynamics of an evolutionary game, by plugging in the payoff matrices A and B. Figure 3.7 shows the directional field for the three games discussed earlier in this chapter. The arrows show the direction of change for different points in the policy space. Since in a 2-action game $x_1 = 1 - x_2$, it suffices to plot only the probability of the first action of each player. The probability with which player 1 plays its first action is plotted on the x-axis, and the probability with which the second player plays its first action is plotted on the y-axis.

For the Prisoners' Dilemma, the field plot shows that all movement is directed towards (0,0), meaning that both players play their second action, which is the Nash equilibrium (Defect, Defect). It is also clear that that this equilibrium is asymptotically stable, since small perturbations of the equilibrium point will lead back to the equilibrium. This shows that Defect is indeed an ESS, as shown in Section 3.2.1.

In the example of the Stag Hunt game, the field plot shows equilibria at (0,0), (1,1) and $(\frac{2}{3}, \frac{2}{3})$, corresponding to the NE (H,H), (S,S), and a mixed equilibrium where both players play S with probability $\frac{2}{3}$. The mixed equilibrium is unstable, as small deviations will lead away from the fixed point. Therefore, the strategy of playing S with probability $\frac{2}{3}$) is not an ESS. It is also clear that (H,H) has a larger basin of attraction than (S,S), which agrees with the fact that this risk dominant equilibrium is the safer choice (see Section 3.1.3).

The field plot for the Battle of the Sexes shows that the mixed equilibrium, where both players play their preferred action with probability $\frac{2}{3}$, is not an ESS. The other two Nash equilibria (O,O) and (F,F) are equally attractive in general, as they have equal basins of attraction. However, the first player prefers (O,O) whereas the second player prefers (F,F).

These examples show that the replicator dynamics are a useful tool to study evolutionary games. The next chapter describes that they are equally useful in describing the behavior of reinforcement learners, by linking evolutionary game theory to reinforcement learning.

Prisoners' Dilemma

$$\begin{array}{cc} & \begin{array}{cc} C & D \end{array} \\ \begin{array}{c} C \\ D \end{array} & \left(\begin{array}{cc} 3,3 & 0,5 \\ 5,0 & 1,1 \end{array} \right) \end{array}$$

Stag Hunt

$$\begin{array}{cc} & \begin{array}{cc} S & H \end{array} \\ \begin{array}{c} S \\ H \end{array} & \left(\begin{array}{cc} 4,4 & 1,3 \\ 3,1 & 3,3 \end{array} \right) \end{array}$$

Battle of the Sexes

$$\begin{array}{cc} & \begin{array}{cc} O & F \end{array} \\ \begin{array}{c} O \\ F \end{array} & \left(\begin{array}{cc} 2,1 & 0,0 \\ 0,0 & 1,2 \end{array} \right) \end{array}$$

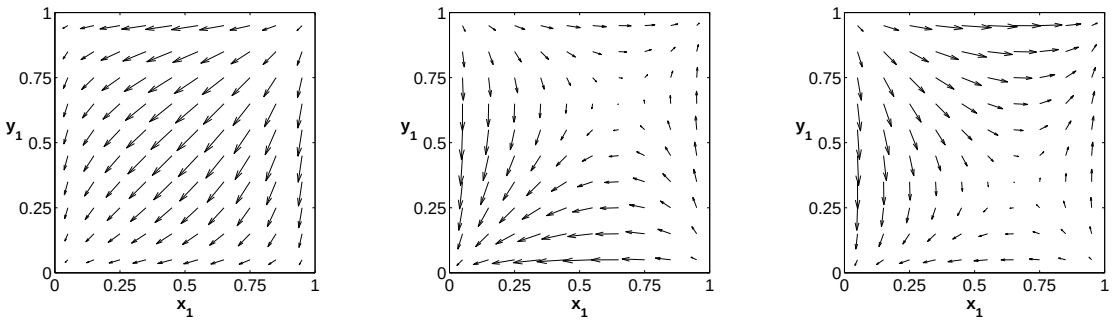


Figure 3.7: Example of the replicator dynamics for three different games, with payoff matrices as defined above. The values on the axis represent the probability with which player 1 (X) and player 2 (Y) play their first action.

Chapter 4

Relating evolutionary game theory and reinforcement learning

This chapter describes the relation between reinforcement learning and evolutionary game theory. This relation was first formalized by Börgers and Sarin (1997), who proved that the continuous time limit of Cross learning, a specific reinforcement learning algorithm, converges to the replicator dynamics. Based on their result, evolutionary models of various reinforcement learning algorithms have been constructed (Tuyls *et al.*, 2006; Panait *et al.*, 2008; Klos *et al.*, 2010).

Section 4.1 introduces the formal relation as described by Börgers and Sarin (1997), and summarizes the models that have been derived for the four RL algorithms considered in this thesis. A recent modification to Q-learning, based on its evolutionary model, is presented in Section 4.2. Furthermore, a new extension of this modification to Lenient Q-learning is proposed. Finally, several advantages of the evolutionary game theoretic approach to reinforcement learning are highlighted.

4.1 Evolutionary dynamics of reinforcement learning

Evolutionary game theory enables the construction and analysis of evolutionary models that describe the dynamics of reinforcement learning. This section presents this relation formally, and describes recently developed evolutionary models of the reinforcement learning algorithms studied in this thesis.

4.1.1 The formal relation: Cross learning

The formal relation between EGT and RL was first established by Börgers and Sarin (1997), based on a RL algorithm known as Cross learning that originated from mathematical psychology. The policy update rule of Cross learning is defined as

$$x_i(t+1) \leftarrow x_i(t)(1 - r_i(t+1)) + r_i(t+1) \tag{4.1}$$
$$\text{if } a_i \text{ is the action taken at time } t \text{ or}$$
$$x_j(t+1) \leftarrow x_j(t)(1 - r_j(t+1)) \tag{4.2}$$
$$\text{for all } a_j \neq a_i$$

where $r_i(t+1) \in [0, 1]$ is the reward given for taking action a_i at time t, as usual. The authors constructed a continuous time limit of the two-player Cross learning model, in which the time interval between two repetitions of the game is infinitesimally small, and the players' adjustment to their policy decreases in size accordingly. They proved that, under the assumption of frequent play and slow movement, this continuous time limit converges to the replicator dynamics of Equations 3.11 and 3.12.

This result proved to be a very important step towards a new theoretical approach to reinforcement learning in multi-agent games. Based on this formal relation, evolutionary models have been developed of various RL algorithms, among which the ones studied in this thesis. These models are discussed in the next sections.

4.1.2 Q-learning

Tuyls *et al.* (2003b) derived an evolutionary model of Q-learning for two-player games, by calculating the continuous time limit of the Boltzmann exploration function. They arrived at the replicator dynamics presented in Equations 4.3 and 4.4. The mathematical details are omitted here, the interested reader is referred to the original publication.

$$\frac{dx_i}{dt} = \frac{\alpha x_i}{\tau}[(Ay)_i - x^T Ay] + x_i \alpha \sum_j x_j ln(\frac{x_j}{x_i}) \tag{4.3}$$

$$\frac{dy_i}{dt} = \frac{\alpha y_i}{\tau}[(Bx)_i - y^T Bx] + y_i \alpha \sum_j y_j ln(\frac{y_j}{y_i}) \tag{4.4}$$

Interesting to note is that the first term of these equations is similar to the replicator dynamics of Equations 3.11 and 3.12, with an additional weight factor α/τ that incorporates the step size and temperature parameter of Q-learning. The first term represents the selection mechanism, that favors an action based on the resulting payoff, relative to the average payoff of the population. The second term introduces mutation, based on the difference in entropy between the action and the population as a whole (Tuyls *et al.*, 2003b). This selection-mutation scheme can be mapped to the exploration-exploitation dilemma, where exploration relates to mutation and exploitation relates to selection. As a result, the evolutionary model provides a different perspective to the exploration-exploitation dilemma.

4.1.3 Lenient Q-learning

Several ways exist in which leniency can be introduced in a learning algorithm, as explained in Section 2.3.2. The most straightforward way to forgive mistakes is to collect several rewards for each action before performing an update step. This update is then based on the highest of those collected rewards. Panait *et al.* (2008) incorporated this modification in the evolutionary model of Q-learning described in the previous section. This modification effects the selection term of the replicator dynamics, since leniency influences the expected reward for an action, as well as the average expected reward of the population as a whole. The expected maximum payoff for action a_i over κ interactions is given by Equations 4.5 and 4.6, for both players. Substituting these for the normal reward matrices A and B in the replicator dynamics leads to the evolutionary model of Lenient Q-learning presented in Equations 4.7 and 4.8.

$$u_i = \sum_j \frac{A_{ij} yj \left[\left(\sum_{k:A_{ik} \leq A_{ij}} y_k \right)^\kappa - \left(\sum_{k:A_{ik} < A_{ij}} y_k \right)^\kappa \right]}{\sum_{k:A_{ik}=A_{ij}} y_k} \tag{4.5}$$

$$w_i = \sum_j \frac{B_{ji} xj \left[\left(\sum_{k:B_{ki} \leq B_{ji}} x_k \right)^\kappa - \left(\sum_{k:B_{ki} < B_{ji}} x_k \right)^\kappa \right]}{\sum_{k:B_{ki}=B_{ji}} x_k} \tag{4.6}$$

$$\frac{dx_i}{dt} = \frac{\alpha x_i}{\tau}(u_i - x^T u) + x_i \alpha \sum_j x_j ln(\frac{x_j}{x_i}) \tag{4.7}$$

$$\frac{dy_i}{dt} = \frac{\alpha y_i}{\tau}(w_i - y^T w) + y_i \alpha \sum_j y_j ln(\frac{y_j}{y_i}) \tag{4.8}$$

Note that only the utilities of the actions for each player have changed compared to Equations 4.3 and 4.4. Where for Q-learning the utility for action a_i is simply the expected payoff given the other player's policy, e.g., $(Ay)_i$ and $(Bx)_i$, for Lenient Q-learning the utility is represented by the expected maximum over κ rewards given by u_i for the first and w_i for the second player.

4.1.4 Learning Automata

This thesis considers the Finite Action-set Learning Automata, as described in Section 2.2.2. Three different variations exist, characterized by the values of the parameters α and β: L_{R-I}, L_{R-P}, and

$L_{R-\epsilon P}$. When applying the L_{R-I} scheme, with $\alpha = 1$ and $\beta = 0$, the update rules of Equations 2.4 and 2.5 reduce to

$$\begin{aligned} x_i(t+1) \leftarrow & x_i(t) + r_i(t+1)(1 - x_i(t)) \\ & = x_i(t)(1 - r_i(t+1)) + r_i(t+1) \qquad (4.9) \\ & \text{if } a_i \text{ is the action taken at time } t \text{ or} \\ x_j(t+1) \leftarrow & x_j(t) - r_i(t+1)x_j(t) \\ & = x_j(t)(1 - r_j(t+1)) \qquad (4.10) \\ & \text{for all } a_j \neq a_i \end{aligned}$$

which equals the Cross learning model of Equations 4.1 and 4.2. Therefore, the relation between Cross learning and the replicator dynamics carries over to learning automata as well, under these conditions (Tuyls *et al.*, 2006). The step size parameter α can be incorporated in the replicator dynamics as a weight factor, which results in the following evolutionary model of the L_{R-I} variation of FALA:

$$\frac{dx_i}{dt} = \alpha x_i[(Ay)_i - x^T Ay] \qquad (4.11)$$

$$\frac{dy_i}{dt} = \alpha y_i[(Bx)_i - y^T Bx] \qquad (4.12)$$

The step size parameter α influences the rate of change of this model without affecting its general behavior. This modification is only necessary for a fair comparison between the different learners, as discussed in Section 5.3.2. This variation of FALA is used throughout the remainder of this thesis.

4.1.5 Regret Minimization

The last RL algorithm considered in this thesis is the polynomial weights variation of regret minimization, described in Section 2.2.3. Recently, an evolutionary model of this variation of regret minimization has been derived, resulting in the following dynamics (Klos *et al.*, 2010):

$$\frac{dx_i}{dt} = \frac{\lambda x_i[(Ay)_i - x^T Ay]}{1 - \lambda[\max_k (Ay)_k - x^T Ay]} \qquad (4.13)$$

$$\frac{dy_i}{dt} = \frac{\lambda y_i[(Bx)_i - y^T Bx]}{1 - \lambda[\max_k (Bx)_k - y^T Bx]} \qquad (4.14)$$

The replicators dynamics of Equations 3.11 and 3.12 can be recognized in the numerator; the denominator represents a weight factor based on the expected loss. Again, the step size parameter of the learning algorithm, in this case λ, determines the rate of change of the dynamical model.

4.2 Recent improvements

Apart from describing the behavior of reinforcement learning algorithms, an evolutionary model can also be used to improve those algorithms (Tuyls *et al.*, 2003a). It has been shown that the actual behavior of Q-learning deviates from the evolutionary prediction based on Equations 4.3 and 4.4, and that the predicted behavior is in fact more desirable (Kaisers and Tuyls, 2010). This led to an improved version of the Q-learning algorithm, called Frequency Adjusted Q-learning (FAQ). This improvement is described in the following section. Furthermore, the same improvement is proposed for Lenient Q-learning in Section 4.2.2.

4.2.1 Frequency adjusted Q-learning

In general, a difference between the expected and actual behavior can occur when the step size of the learning is large (Börgers and Sarin, 1997). However, Kaisers and Tuyls (2010) show that the discrepancies observed in Q-learning remain when the step size is decreased. They argue that the cause of this difference

by calculating the derivative of the action-value update function, and show that it differs by a factor x_i from the derivative used to construct the evolutionary model of Q-learning. This is caused by the fact that the action-values in Q-learning are updated asynchronously: the value of an action is only updated when it is selected. The evolutionary model, on the other hand, was derived under the assumption that all actions are updated equally often (Tuyls *et al.*, 2003b).

Kaisers and Tuyls (2010) further argue that the predicted behavior is more desirable than the actual behavior of Q-learning. This lead them to construct an improved version of Q-learning, in which the action-value update is weighted inversely proportional to the probability with which the action is selected:

$$Q_{a_i}(t+1) \leftarrow Q_{a_i}(t) + \frac{1}{x_i}\alpha\left[r(t+1) + \gamma \max_j Q_{a_j}(t) - Q_{a_i}(t)\right] \tag{4.15}$$

However, this functions is only valid in the infinitesimal limit of α, as otherwise the fraction α/x_i may become larger than 1. This would violate the assumptions under which the algorithm converges (Watkins and Dayan, 1992). In order for the algorithm to be numerically applicable, the authors defined a generalized version of the frequency adjusted Q-learning algorithm, by introducing a variable $\beta \in [0, 1)$:

$$Q_{a_i}(t+1) \leftarrow Q_{a_i}(t) + \min\left(\frac{\beta}{x_i}, 1\right)\alpha\left[r(t+1) + \gamma \max_j Q_{a_j}(t) - Q_{a_i}(t)\right] \tag{4.16}$$

When $\beta = 1$, FAQ reduces to normal Q-learning; therefore this value is excluded from the allowed range of β. Kaisers and Tuyls (2010) further show that the value of β controls the area of the policy space for which FAQ is valid: if $x_i \geq \beta$, FAQ behaves optimally; if $x_i < \beta$, FAQ reduces to regular Q-learning. This is visualized in Figure 4.1, which shows the area of the reduced policy space (see Section 5.4) for which FAQ is valid, based on the value of β.

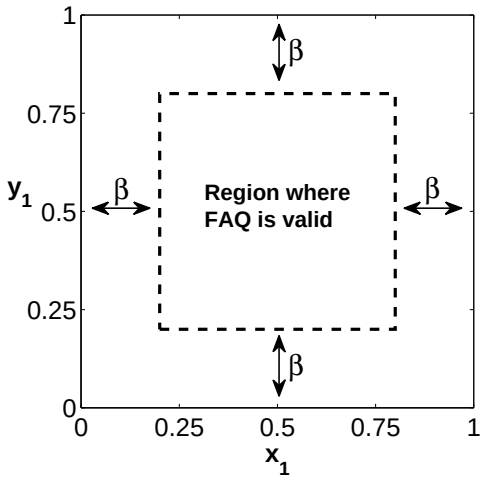


Figure 4.1: The area of the policy space for which FAQ is valid, based on the value of β.

Theoretically, this means the the range of β can be further narrowed, since it reduces the policy space from both sides: $0 < \beta < \frac{1}{2}$. Practically, β should be chosen as small as possible to ensure the validity of FAQ for a large part of the policy space. Section 5.2.2 describes a different way of ensuring the validity of FAQ, by using the Boltzmann exploration mechanism to reduce the reachable policy space.

Kaisers and Tuyls (2010) show empirically that the behavior of FAQ matches the evolutionary model that was originally designed for Q-learning, whereas Q-learning itself deviates from it. Furthermore, they show that FAQ is less sensitive to the initialization of the Q-values, whereas Q-learning behaves differently depending on the initial action-values. The latter fact makes FAQ a robust choice for many applications where correct initialization might be impossible.

4.2.2 Lenient frequency adjusted Q-learning

Since the action-value update rule for Lenient Q-learning is equal to that of normal Q-learning, its behavior is similarly influenced by the asynchronous nature of the update rule. However, the evolutionary

model of Lenient Q-learning, presented in Section 4.1.3, is based directly on the evolutionary model of Q-learning (Panait *et al.*, 2008). Therefore, the same discrepancies between predicted behavior and actual behavior are expected. Figure 4.2 shows an example of the actual behavior of LQ (solid lines) as compared to the evolutionary model (arrows) in the Stag Hunt game. Clearly, the actual behavior differs from its expectation. Not only are the trajectories shaped differently, in several cases the learning process even converges to a different equilibrium than expected.

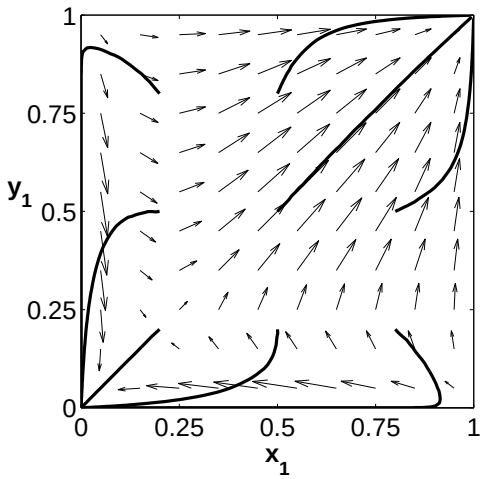


Figure 4.2: Example of the discrepancy between the actual behavior of Lenient Q-learning and the evolutionary model.

This thesis proposes the Lenient Frequency Adjusted Q-learning (LFAQ) algorithm that makes use of the same improvement as FAQ, described in Section 4.2.1. The action-value update rule of LFAQ is equal to that of FAQ; the difference is again that the lenient version collects κ rewards before updating its Q-values based on the highest of those rewards. Section 6.1 provides empirical proof that LFAQ indeed matches the evolutionary model proposed by Panait *et al.* (2008).

4.3 Advantages of this approach

The evolutionary game theoretic approach to reinforcement learning, as described in this chapter, offers some interesting insights into the inner working of reinforcement learning algorithms. Typically, when applied to the area of multi-agent systems these algorithms are *blackboxed* in nature. Especially when the number of agents and actions increases, the complexity of the learning problem prevents an easy understanding of how a learner behaves, and why it behaves the way it does.

The replicator dynamics offer a solution to this problem. By deriving an evolutionary model of a reinforcement learning algorithm, it becomes much easier to analyze and understand its behavior. It is possible to predict the behavior and convergence of a learning process in advance, and easily compare how different learners fare in the problem at hand. As a result, the EGT approach can also guide the otherwise tedious task of parameter tuning, by allowing to analyze the effect that the various parameters have on the learning behavior in advance.

Furthermore, this approach might also lead to the construction of new reinforcement learning algorithms that are suited to solve specific problems. It has already been shown how the evolutionary model of Q-learning led to an improvement of that learning algorithm. In a similar way, it is possible to define preferred dynamics of a learning algorithm in advance by constructing an evolutionary model. Working backwards, a learning algorithm can be derived that matches the preferred behavior. Tuyls *et al.* (2003a) show how this procedure can be used to derive a learning algorithm that guarantees convergence in all 2×2 normal form games.

These advantages highlight the strengths of the EGT approach to RL. Not only can this approach lead to better insights into the behavior of RL algorithms, it can also point towards the creation of new RL algorithms that are tailored to specific problems.

Chapter 5

Methodology

The main goal of the experiments is to get an insight into the behavior and performance of various reinforcement learning algorithms. This insight is acquired in two ways. The first approach is to analyze the policy trajectories of the learners, by running simulations with different starting points (initial policies) and tracking how the learners' policies evolve. These trajectories can be used to analyze various characteristics of the learner, such as its behavior in different regions of the policy space, its learning speed and the equilibria it might converge to. The second approach is to analyze learning behavior by looking at the theoretical model of the learner; i.e., by looking at the associated evolutionary dynamics.

This chapter outlines the experimental setup of this thesis. A brief introduction to the different games that are used as learning environment is given in Section 5.1. Each RL algorithm has various parameters that need to be fine-tuned. These are described in Section 5.2. Section 5.3 discusses the experimental setup, in particular the distinction between self play and mixed play. Finally, the tools used to analyze the behavior, convergence properties and performance of the learners are detailed.

5.1 Testbed for the experiments

This section discusses the five games that are used as a testbed for the learning algorithms. All are 2×2 normal form games, meaning that they are two-player games in which each player has to choose between two actions. Several of these have already been introduced in Chapter 3.

Three distinct classes of 2×2 normal form games can be identified (Gintis, 2009). The first class consists of games with one pure Nash equilibrium, such as the Prisoners' Dilemma. The second class of games has two pure and one mixed Nash equilibrium. An example of this class of games is the Battle of the Sexes. Finally, the third class of games has only one mixed Nash equilibrium. An example of this last class is the Matching Pennies game, which is introduced later in this section.

5.1.1 Games under consideration

Prisoners' Dilemma

The Prisoners' Dilemma (PD) has already been extensively studied in Chapter 3, for a detailed description see Section 3.1.1. For convenience, the payoff matrix is presented again in Figure 5.1. The game's only Nash equilibrium (NE) is the strategy pair (D,D), leading to a reward of 1 for both players. The dilemma in this case it that this NE is not Pareto optimal, since both players would be better off playing (C,C). The strategy D is also an evolutionarily stable strategy (ESS), as explained in Section 3.2.1.

$$\begin{array}{c} \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cc} C & D \end{array} \\ \begin{pmatrix} 3,3 & 0,5 \\ 5,0 & 1,1 \end{pmatrix} \end{array}$$

Figure 5.1: Payoff matrix for the Prisoners' Dilemma, with the actions Cooperate (C) and Defect (D).

Battle of the Sexes

The Battle of the Sexes (see Section 3.1.2) is an example of the second class of games; its payoff matrix is given in Figure 5.2. The game is characterized by two pure NE, (O,O) and (F,F), and one mixed NE where each player plays its preferred action with probability $\frac{2}{3}$. The two pure equilibria are both also ESS, the mixed equilibrium however is not. Furthermore, the two pure NE are not equally preferable by both players: player 1 prefers (O,O), and player 2 prefers (F,F).

$$\begin{array}{c} \\ O \\ F \end{array} \begin{array}{c} \begin{array}{cc} O & F \end{array} \\ \begin{pmatrix} 2,1 & 0,0 \\ 0,0 & 1,2 \end{pmatrix} \end{array}$$

Figure 5.2: Payoff matrix for the Battle of the Sexes, with the actions Opera (O) and Fight (F).

Stag Hunt Game

The Stag Hunt game, described in Section 3.1.3, also belongs to the second class of games. Its payoff matrix is given in Figure 5.3. It has characteristics similar to the Battle of the Sexes: two pure NE, (S,S) and (H,H), and one mixed NE where each player plays S with probability $\frac{2}{3}$. However, in this case both players benefit most from playing (S,S), which is the Pareto optimal (or payoff dominant) equilibrium. On the other hand, H is a risk dominant strategy since the payoff for playing H does not depend on the action of the opponent. Therefore, also (H,H) has a considerable basin of attraction (see also Figure 3.7).

$$\begin{array}{c} \\ S \\ H \end{array} \begin{array}{c} \begin{array}{cc} S & H \end{array} \\ \begin{pmatrix} 4,4 & 1,3 \\ 3,1 & 3,3 \end{pmatrix} \end{array}$$

Figure 5.3: Payoff matrix for the Stag Hunt game, with the actions Stag (S) and Hare (H).

Coordination Game

The Coordination Game is a modification of the Battle of the Sexes. Where in the latter both players prefer a different equilibrium, in the Coordination Game they agree that going to the Opera is most enjoyable. This is expressed in the payoff matrix as shown in Figure 5.4. The two pure NE are still (O,O) and (F,F), but in this case (O,O) Pareto dominates (F,F). The mixed NE is that both players play O with probability $\frac{1}{3}$. Similar to the Battle of the Sexes, both pure equilibria are also ESS, whereas the mixed equilibrium is not.

$$\begin{array}{c} \\ O \\ F \end{array} \begin{array}{c} \begin{array}{cc} O & F \end{array} \\ \begin{pmatrix} 2,2 & 0,0 \\ 0,0 & 1,1 \end{pmatrix} \end{array}$$

Figure 5.4: Payoff matrix for the Coordination Game, with the actions Opera (O) and Fight (F).

Matching Pennies

The last game under consideration is the Matching Pennies game. In this game, both players hold a coin of one Euro (or, traditionally, a Pennie). They independently choose to show their coin with either its head or its tail on top. If both players show a different side, player 1 gets both coins. If they show the same side, player 2 gets both coins. The payoff matrix is shown in Figure 5.5. This is an example of the third class of games, in which there is only one mixed Nash equilibrium: both players play each action with probability $\frac{1}{2}$.

Interesting in this case is that this mixed equilibrium is not evolutionarily stable. This can be seen by looking at the corresponding replicator dynamics, shown in Figure 5.6. These dynamics are obtained by plugging in the game's payoff matrices in Equations 3.11 and 3.12. The figure shows that the trajectories

$$\begin{array}{cc} & \begin{array}{cc} H & T \end{array} \\ \begin{array}{c} H \\ T \end{array} & \left(\begin{array}{cc} -1,1 & 1,-1 \\ 1,-1 & -1,1 \end{array} \right) \end{array}$$

Figure 5.5: Payoff matrix for the Matching Pennies game, with the actions Head (H) and Tail (T).

of dynamical system are closed orbits around the equilibrium, which violates the definition of asymptotic stability (Definition 8). Therefore, the equilibrium is not an ESS (Property 3). However, the equilibrium is neutrally stable, since each trajectory that starts sufficiently close to the equilibrium remains arbitrarily close to it.

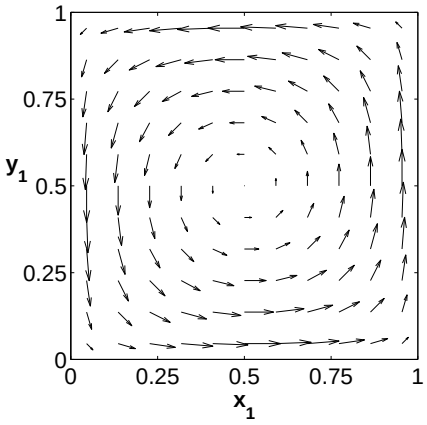


Figure 5.6: Replicator dynamics of the Matching Pennies game.

5.1.2 Normalization

The payoffs for the different games described in the previous section are not limited to a specific range of values. However, the value of the reward signal used by the RL algorithms can make a difference. Some learners, such as FALA and RM, require that the rewards lie in the range $[0, 1]$, as can be seen by the formulation of their respective update functions. Therefore, normalization is required in order to convert the payoffs to this range. For the variants of Q-learning presented in this thesis, the payoff values are not limited to a specific range. Normalization can be applied in this case nevertheless, since the Boltzmann action selection mechanism is insensitive to translation or multiplication of the Q-values, and the Q-values in turn depend linearly on the payoffs received. As a result normalization has no influence on the behavior of Q-learning.

Therefore, in order to ensure a fair competition, all game matrices are normalized before they are used in an experiment. The normalization ensures that all payoff values are in the range $[0, 1]$. Figure 5.7 shows the normalized payoff matrices for the five games discussed in the previous section.

These normalized payoff matrices will be used in all experiments, as well as in the discussion of the results. For example, in case a comparison is made between the reward earned by a learner and the expected reward for a certain Nash equilibrium, implicitly the expected normalized reward is used.

5.2 Parameter settings

An important aspect of applying reinforcement learning algorithms, or most types of learning algorithms for that matter, is parameter tuning. Each learning algorithm considered here has one or more parameters that influence its behavior and that need to be set right in order for it to work properly. Some of these parameters are unique to a specific learning algorithm, others are more general. For example, most algorithms use a parameter to control the step size, or learning rate, of the learning process. Controlling the learning rate is an important issue, as it can greatly influence the behavior of the learner.

$$\begin{array}{c} \\ C \\ D \end{array}\begin{array}{c} \begin{array}{cc} C & D \end{array} \\ \begin{pmatrix} \frac{3}{5},\frac{3}{5} & 0,1 \\ 1,0 & \frac{1}{5},\frac{1}{5} \end{pmatrix} \end{array}$$

Prisoners' Dilemma

$$\begin{array}{c} \\ O \\ F \end{array}\begin{array}{c} \begin{array}{cc} O & F \end{array} \\ \begin{pmatrix} 1,\frac{1}{2} & 0,0 \\ 0,0 & \frac{1}{2},1 \end{pmatrix} \end{array}$$

Battle of the Sexes

$$\begin{array}{c} \\ S \\ H \end{array}\begin{array}{c} \begin{array}{cc} S & H \end{array} \\ \begin{pmatrix} 1,1 & 0,\frac{2}{3} \\ \frac{2}{3},0 & \frac{2}{3},\frac{2}{3} \end{pmatrix} \end{array}$$

Stag Hunt

$$\begin{array}{c} \\ O \\ F \end{array}\begin{array}{c} \begin{array}{cc} O & F \end{array} \\ \begin{pmatrix} 1,1 & 0,0 \\ 0,0 & \frac{1}{2},\frac{1}{2} \end{pmatrix} \end{array}$$

Coordination Game

$$\begin{array}{c} \\ H \\ T \end{array}\begin{array}{c} \begin{array}{cc} H & T \end{array} \\ \begin{pmatrix} 0,1 & 1,0 \\ 1,0 & 0,1 \end{pmatrix} \end{array}$$

Matching Pennies

Figure 5.7: Normalized payoff matrices for the five different games.

This section provides an overview of the various parameter settings used in this thesis. First, the parameters controlling the learning rate are discussed in general, as well as in relation to the various RL algorithms. Next, the more specific parameters that are unique to each learning algorithm are described.

5.2.1 General parameters

Each RL algorithm used in this thesis has some form of weight parameter that controls the influence that the received reward has on the update of the policy or value function. This weight parameter can thus be used to control the rate of change, or learning rate, of the learner. Increasing the learning rate speeds up the learning process and might therefore lead to faster convergence. However, a larger learning rate makes the learner respond more drastically to each reward update, which can in the end destabilize the learning process. Eventually, the learner might not converge at all. A very small learning rate, on the other hand, leads to smooth and predictable learning behavior. The time needed to converge can become very long though, and whereas such a smooth and predictable learning trajectory can be interesting theoretically it does not always reflect real-life situations since learning in a dynamic and changing environment often needs to be quick. Therefore it is important to find the right balance between a quick learning process on the one hand, and predictable behavior on the other.

Note that there is a difference between the terms learning *rate* and learning *speed*. The learning rate, or step size, indicates the combination of parameter settings that control the size of the update step of a learner. The learning speed indicates the size of the resulting policy change.

An overview describing the individual step size parameters of each RL algorithm is given in Table 5.1.

5.2.2 Algorithm specific parameters

Where the FALA and RM algorithms only use a step size parameter, the two Q-learning based algorithms FAQ and LFAQ require some more fine tuning. They both share two additional variables: the discount factor for future rewards γ and the temperature τ for the Boltzmann exploration mechanism. Furthermore, LFAQ has a leniency parameter κ that controls the number of samples taken before doing an update step.

Discount factor

The discount factor γ of Q-learning controls the importance of future rewards. If $\gamma = 0$, the agent is short sighted and will only consider immediate rewards. When $\gamma = 1$, it is far-sighted and does not distinguish between immediate and future rewards, given that no time related penalties are applied. Both extremes are in general not favorable; the former ignores valuable information about what lies ahead, and the latter can be inefficient since it does not favor the shortest path over any solution that eventually leads to the goal.

In single-stage games such as the ones under consideration, the future reward is independent of the action chosen and therefore the discount factor plays a lesser role. Suppose action i is chosen at time t. The action-value function of Q-learning (Equation 2.2) can now be written as

$$Q_i(t+1) \leftarrow Q_i(t) + \alpha\left[r(t+1) - Q_i(t)\right] + f(t) \tag{5.1}$$

Frequency Adjusted Q-learning

α, β The learning speed of FAQ-learning is not only dependent on step size parameters but changes throughout the learning process, depending on the current policy (Kaisers and Tuyls, 2010). As Equation 4.16 shows, the size of an action-value update is inversely related to the probability x_i with which this action is selected. The parameters $\alpha, \beta \in [0, 1]$ set the bounds between which this variation occurs. In the equation, the learning rate is given by $min\left(\frac{\beta}{x_i}, 1\right)\alpha$. This means that when the action selection probability increases, the learning rate decreases towards $\alpha\beta$; when the action selection probability decreases, the learning rate increases towards α. Therefore, α and $\alpha\beta$ are respectively the upper and lower bound of the learning rate.

Lenient FAQ-learning

α, β, κ Lenient-FAQ learning has the same step size parameters as FAQ-learning. However, the degree of leniency, κ, also plays a role. Because the learner collects κ rewards before performing a single update step, the actual learning speed is κ times smaller than that of FAQ when using the same values for α and β.

Learning Automata

α, β Learning automata (FALA) also use step size parameters α and β, be it with a different meaning. As shown in Equations 2.4 and 2.5, α sets the effect that the reward r has on the update step whereas β controls the effect of the penalty, which is defined as $(1 - r)$ where 1 is assumed to be the maximum possible reward. In this thesis, only the L_{R-I} update scheme is used, where $\beta = 0$, to assure that the learning process matches the RD model of Cross Learning (see Section 4.1.4). Therefore, in this case only α is used to influence the learning rate.

Regret Minimization

λ The learning rate of the Polynomial Weights RM algorithm is controlled by a single step size parameter λ. This parameter defines the effect that an action's loss l has on its concurrent weight update (see Equation 2.6). It can therefore be seen as an analog of the parameter α used in the other three learning algorithms, as it also determines the size of the update steps of the learner.

Table 5.1: Overview of the step size parameters of each learning algorithm.

where $f(t)$ is the expected future reward at time t, defined as

$$f(t) = \alpha\gamma \max_j Q_j(t). \tag{5.2}$$

Rewriting the Boltzmann equation (2.3) for two-action games, separating the future reward f from the action-value Q for action i gives

$$x_i = \frac{e^{(Q_i+f)\tau^{-1}}}{e^{(Q_i+f)\tau^{-1}} + e^{(Q_j+g)\tau^{-1}}} \tag{5.3}$$

where g is the expected future reward at the time when action j was last selected. Equation 5.3 can be rewritten as

$$x_i = \frac{e^{Q_i\tau^{-1}} \cdot e^{f\tau^{-1}}}{e^{Q_i\tau^{-1}} \cdot e^{f\tau^{-1}} + e^{Q_j\tau^{-1}} \cdot e^{g\tau^{-1}}}. \tag{5.4}$$

Since the actions are not updated simultaneously, in general $f \neq g$ and as a result the policy update does depend on the future reward and therefore on γ. However, in the limit when $\alpha \to 0$, $f \approx g$ and the future reward cancels out in Equation 5.4, making the policy update independent of γ.

Exploration temperature

Another important parameter of the Q-learning algorithms under consideration is the temperature τ that, through the Boltzmann exploration mechanism of Equation 2.3, controls the balance between exploration and exploitation. A high temperature promotes exploration whereas a low temperature favours exploitation. As a result, τ also controls the area of the policy space that is reachable by the learning process, since it places a bound on the minimum action selection probability that is possible. Following Equation 2.3, for a single-stage two-action game,

$$x_{min} = \frac{e^{Q_{min} \cdot \tau^{-1}}}{e^{Q_{min} \cdot \tau^{-1}} + e^{Q_{max} \cdot \tau^{-1}}}. \tag{5.5}$$

The value of Q_{min}, and similarly of Q_{max}, can be calculated by the following formula (Kaisers and Tuyls, 2010):

$$Q_{min} = \sum_{t=0}^{\infty} \gamma^t r_{min} = \frac{1}{1-\gamma} r_{min}. \tag{5.6}$$

For discount factor $\gamma = 0$ and rewards $r \in [0, 1]$ this leads to $Q_{min} = 0$ and $Q_{max} = 1$, and as a result

$$x_{min} = \frac{1}{1 + e^{\tau^{-1}}}. \tag{5.7}$$

This insight can be used in FAQ and LFAQ to select β in such a way that $x_i \geq \beta$ is guaranteed, which ensures the validity of (L)FAQ for the whole reachable policy space, as explained in Section 4.2.1.

Leniency

The degree of leniency κ in LFAQ sets the number of rewards that are collected before an update step is performed based on the highest value of those rewards. The higher κ, the more lenient, or forgiving, an agent is towards possible mistakes made by the other agents in a cooperative game. When $\kappa = 1$, LFAQ reduces to regular FAQ. Panait *et al.* (2008) investigate the behavior of lenient Q-learning for different settings of κ and show that leniency indeed improves the convergence properties. In accordance with their results, this thesis uses $\kappa = 5$ unless otherwise specified.

5.3 Experimental setup

This section introduces the experiments that are conducted in this thesis. The general outline of the main experiments is given, as well as of the more specific experiments to fine-tune the learning rate. The next section presents the analytical tools that are used to present the results.

5.3.1 Self play versus mixed play

A distinction can be made between experiments that involve two agents using the same learning algorithm, and those that involve agents using different learning algorithms. The first case of homogeneous or self play experiments serves as a base line for the more advanced heterogeneous, or mixed play, experiments. All analysis tools described in this section can be used for either form of experiments.

The self play experiments analyze homogeneous pairs of the four different learning algorithms FAQ, LFAQ, FALA and RM in the five different games that are described in Section 5.1.1. This amounts to a total of $4 \times 5 = 20$ different situations.

The mixed play experiments are more extensive, with pairwise coupling of the different learning algorithms. The self play results for each learner are used as a base line in order to compare the different combinations of learners.

5.3.2 Fine-tuning the learning rate

Analyzing the learning speed of the various RL algorithms is important in two ways. First of all, the learning rate needs to be tuned in such a way that the learners behave reliably, without slowing down the learning process more than necessary. Secondly, when two different learners oppose each other in a

single game it might be that they do not learn equally fast. This can lead to artifacts caused by the mere difference in learning speed rather then a true differentiation in qualitative learning dynamics. Therefore, it is necessary to analyze how the different step size parameters relate, i.e., which settings lead to similar learning speeds for different learners, in order to elicit the underlying qualitative differences.

The first part of the analysis, i.e., finding the balance between speed and predictability, is achieved by means of two different analytical tools. The first one shows the convergence properties of the learning process over the policy space, by calculating the region of the policy space for which convergence is guaranteed. Doing this for various learning rates gives an insight into the reliability of the convergence of the learning process for those specific settings. The second tool shows the behavior of the learners, looking in more detail at the predictability of the actual learning trajectories. Sections 5.4.1 and 5.4.2 describe these analytical tools in detail. Together they can be used to justify which step size gives reliable experiments.

The second part, relating the learning rate to the actual learning speed, is achieved by measuring the average time to convergence for the different algorithms and settings. Although the learning speed might vary over the trajectory, the time needed to reach an equilibrium still gives a good indication of the overall learning speed of an algorithm. These results are used to match the learning speeds of the different learners, thereby providing a fair competition in the heterogeneous learning experiments.

5.4 Analytical tools

In order to investigate the behavior, convergence properties and performance of the different RL algorithms, experiments are conducted with various parameter settings and initial policy profiles. This section outlines in detail the various analytical tools used in these experiments. These tools are grouped by their goal, i.e., whether they put emphasis on the behavior, the convergence properties, or the performance of the learners.

An important aspect of the games under consideration is the fact that they are all two-player games with two actions for each player. Therefore, the action selection probabilities that make up an agent's policy are directly linked by the fact that $\sum_i x_i = 1$ and therefore, with only two actions, $x_1 = 1 - x_2$ and vice versa. This makes it possible to reduce the policy pair (x, y) to (x_1, y_1) without loss of information. This *reduced policy pair* can easily be plotted in the unit square, making it easy to analyze policy trajectories, for example. Plotting this reduced policy pair is an important aspect of most of the tools described in the remainder of the section.

5.4.1 behavior analysis

Insight into the behavior of the learners can best be acquired by looking at how the learner's policies evolve over time. This can be done both by simulating the learning process, and by analyzing the corresponding replicator dynamics. This section describes several tools that are used to present the results.

Policy trajectory plot

A policy trajectory plot can be created by recording the agents' changing policies over time, and subsequently plotting $x_1(t)$ against $y_1(t)$ in the reduced policy space as described above. This can be done for different initial policies in order to get a better insight into the behavior of the learners in different areas of the policy space. With increasing step size these trajectories become less certain, in which case the average trajectory of several simulations is plotted. Which situation applies is indicated when describing the experiment.

Standard deviation plot

The policy trajectory plot shows either a single trajectory or the average over multiple simulation runs. In order to get a better idea about the uncertainty of the trajectories the standard deviation over multiple runs with the same initial policy can be calculated and plotted. The standard deviation σ of a given time

step t on the trajectory is calculated as

$$\sigma(t) = \sqrt{\sum_{r=1}^{R} \frac{[x_1^r(t) - \mu_{x_1}(t)]^2 + [y_1^r(t) - \mu_{y_1}(t)]^2}{R}} \tag{5.8}$$

where R is the number of runs, $x_1^r(t)$ is the value of the r^{th} run of x_1 at time t, and $\mu_{x_1}(t)$ is the average value of x_1 at that point. The same goes for y. The standard deviation plot can for example be used to compare the (un)certainty of the learning process for various step sizes.

Convergence speed comparison

Convergence speed plots are used to compare the learning speed of various algorithms over time. At each time step t, the Euclidean distance d of the policy pair $(x_1(t), y_1(t))$ to the equilibrium (x^*, y^*) is calculated by

$$d(t) = \sqrt{(x_1(t) - x_1^*)^2 + (y_1(t) - y_1^*)}^2. \tag{5.9}$$

Now, setting one learner as base line, the relative speed of the other learners can be plotted as the difference between their distance to the equilibrium and that of the base line learner. Comparing the convergence speeds in this way shows whether the learners behave differently throughout the learning process; it might for example be that a learner is relatively fast in the beginning of the process, but slow at the end. Such insights can be valuable when deciding on a learning algorithm for a certain application domain.

Directional field of the replicator dynamics

The directional field plot of the replicator dynamics is the theoretical model to which the individual simulated policy trajectories converge in the limit. Therefore, it is a valuable tool to assess whether the learning process behaves as expected. Furthermore, it can be used to fine tune some of the learning algorithm's parameters in advance, before the actual simulation is carried out. For example, the effect of the temperature τ on the behavior of FAQ and LFAQ is immediately visible through the replicator dynamics.

Plotting the RD field is done in a way similar to the policy trajectory plots, be making use of the reduced policy space. The direction of the field is plotted as uniformly spaced arrows, their length indicating the rate of change. The similarity of the RD plot and the policy trajectory plot also makes it possible to plot both at the same time, giving an even better idea whether the learners behave as expected. This latter application is especially useful when validating the LFAQ algorithm in Section 6.1.

5.4.2 Convergence properties

Next to the general behavior of the learning algorithms, the convergence properties are also studied in detail. When do the learners converge, where do they converge to for a specific initial policy, and how long does it take? Three main tools are used to study these properties.

ϵ-Convergence

The most important question regarding convergence is: when does the learning process converge? The most simple answer is to measure at what point the policy pair of the learners has converged to the (Nash) equilibrium of the game. However, exact convergence to the equilibrium can take a very long time, if it happens at all. This is do to the fact that each algorithm's learning rate, by way of construction, slows down as it approaches the equilibrium. Therefore, instead of exact convergence the concept of ϵ-*convergence* is used. In order to use this concept, a measure for the distance between two policies is needed. In accordance with Van den Herik *et al.* (2007) the distance between policy x and a fixed policy y is defined as

$$d(x, y) = \max_i |x_i - y_i|. \tag{5.10}$$

Then, a policy x is considered to have ϵ-*converged* to y at time step T if

$$d(x(t), y) < \epsilon, \forall t \geq T. \tag{5.11}$$

Consequently, the learning process is considered to have ϵ-*converged* to an equilibrium if all learners' policies have ϵ-*converged* to this equilibrium.

This measure of convergence is chosen for its intuitive meaning: a learning process has converged if all agents' action probabilities deviate less than ϵ from their desired value. It is used to determine to which equilibrium the learners converge, and at what time step T they do so.

Convergence pureness plot

Using the aforementioned convergence measure, it is possible to construct a gradient field showing the convergence pureness of the learning process over the policy space. This gradient field is constructed by dividing the policy space in a grid, and simulating the learning process starting at each grid cell. By running several simulations for each cell, with uniformly distributed starting points, the percentage-wise convergence to the different equilibria can be determined. The cell's pureness p is defined as

$$p = \frac{\max_i c_i}{\sum_j c_j} \tag{5.12}$$

where c_i is the number of simulation runs that converged to the i^{th} equilibrium of the game.

To achieve a high resolution gradient field, a large number of simulations is required. To partially overcome this problem, the field can be gradually refined. Starting with a low resolution, the whole field is constructed once. Next, the resolution is increased, but only in the areas that where uncertain in the previous step. Especially when the learning process is relatively pure, or certain, the number of simulations required to achieve a reasonable resolution is significantly reduced by this gradual refinement method.

The convergence pureness plot is particularly useful for fine tuning the learning rate as explained in Section 5.2.1. Gradient fields can be constructed for different step sizes to show at which point a satisfactory level of certainty is reached, by calculating the percentage of the policy space for which convergence is certain.

Basin of attraction

As defined in Section 3.2.3, the basin of attraction of an equilibrium is the collection of points through which the learning trajectory will eventually converge to that equilibrium. The directional field plot of the replicator dynamics already indicates where each basin of attraction might be located, by looking at the arrows pointing towards an equilibrium. However, this visual analysis is limited to the number of arrows that can reasonably be plotted, and therefore a more exact method is required.

Using the replicator dynamics it is possible to calculate the basins of attraction in a given game with arbitrary precision. First, the policy space is divided in a grid with a certain resolution, say 100×100. This resolution can be chosen arbitrarily, depending on the required accuracy and the available computation time. For simplicity, each grid cell is represented as its center point only. Next, for each grid point the direction of the replicator dynamics is calculated. These directions are then used to calculate a path, from each grid point, to some equilibrium. The starting cell of this path is then assigned to the basin of attraction of that particular equilibrium. This procedure can be carried out incrementally, in which case a path ends at the first cell for which the basin of attraction is already known. It is also possible that a path never ends at an equilibrium. In that case, at some point the same cell will be visited twice, and all cells in that path are excluded from any basin of attraction.

In the end, all grid cells are assigned either to a basin of attraction, or excluded from it. The basins of attraction can then be visualized by plotting their border, and the percentage of the policy space lying within a basin of attraction can be calculated easily.

5.4.3 Performance analysis

The performance of a RL algorithm is by definition related to the reward received during game play, as the learner's goal is simply to maximize its reward over time. Therefore, the most straightforward way to measure a learner's performance is by looking at its cumulative reward.

Average reward plot

The cumulative reward R of a learner can be calculated by summing all consecutive rewards received during the learning process. However, is is often more interesting to look at the average reward $\bar{R}$ over time, where $\bar{R}$ at time T is defined by

$$\bar{R}(T) = \frac{\sum_{t=1}^{T} r(t)}{T}. \tag{5.13}$$

Plotting the average reward over time gives an insight into how quick the learner improves its policy and at what point it reaches an equilibrium. It can also be used to compare different learners qualitatively; especially in the case of mixed learners it can be used to show which learner is more efficient before reaching equilibrium.

Chapter 6

Results

As indicated in Chapter 5, several different sets of experiments are performed in order to analyze the behavior, convergence properties en performance of the reinforcement learning algorithms under consideration. This chapter presents the results.

The remainder of this chapter is divided in four parts. The first part, Section 6.1, highlights the difference between regular Lenient Q-learning and the newly proposed Lenient Frequency Adjusted Q-learning, and demonstrates that the latter perfectly adheres to the evolutionary predictions whereas LQ may deviate considerably. Section 6.2 is devoted to fine tuning the step size parameters of the different learning algorithms. This fine tuning is needed in order to ensure reliable results and a fair competition, especially in a heterogeneous setting. Sections 6.3 and 6.4 present the homogeneous (self play) and heterogeneous (mixed play) analysis of the different learning algorithms, respectively. The chapter concludes with a summary of the main results.

6.1 Validating Lenient FAQ-learning

In this section, Lenient Q-learning and Lenient FAQ-learning are compared with the evolutionary model derived by Panait *et al.* (2008), in order to investigate whether frequency adjustment indeed resolves the discrepancies observed between the predicted and actual behavior of Lenient Q-learning. Furthermore, the advantage of leniency in cooperative games is illustrated by an example, showing that an increase in leniency raises the probability of converging to the Pareto optimal equilibrium.

6.1.1 Comparing LQ, LFAQ, and the evolutionary model

The behavior of Lenient Q-learning (LQ) and Lenient FAQ-learning (LFAQ) are compared, in order to investigate empirically whether LFAQ better matches the behavior predicted by its replicator dynamics. This is done by comparing the learners' policy trajectories with the directional field of the evolutionary model in various games. As explained in Section 4.2.2, the policy trajectories of LQ deviate from their expected path in a way similar to those of Q-learning. By introducing an extra term in the action-value update rule that compensates for the frequency with which an action is chosen, LFAQ aims to counter this deviation.

In order to be comparable to Kaisers and Tuyls (2010), who investigate the discrepancy in the case of Q-learning, a similar setup is used for the experiments. Three games are selected that represent different classes: the Prisoners' Dilemma with one pure Nash equilibrium; the Stag Hunt with two stable pure Nash equilibria and one mixed equilibrium; and the Matching Pennies game with one mixed Nash equilibrium. Of particular interest is the Stag Hunt, which has one pay-off dominant equilibrium and one risk-dominant equilibrium. The concept of leniency was introduced specifically to improve convergence to such a pay-off dominant Nash equilibrium in cooperative games.

As observed by Kaisers and Tuyls (2010), different initializations of the Q-values result in differences in the learning behavior of Q-learning. Since LQ is a direct extension of Q-learning, a similar effect is expected. Therefore, experiments include different initializations of the Q-values, based on the minimum (pessimistic), mean (neutral), and maximum (optimistic) possible Q-values given the game's reward space. The minimum, and similarly the mean and maximum value, can be found using Equation 5.6.

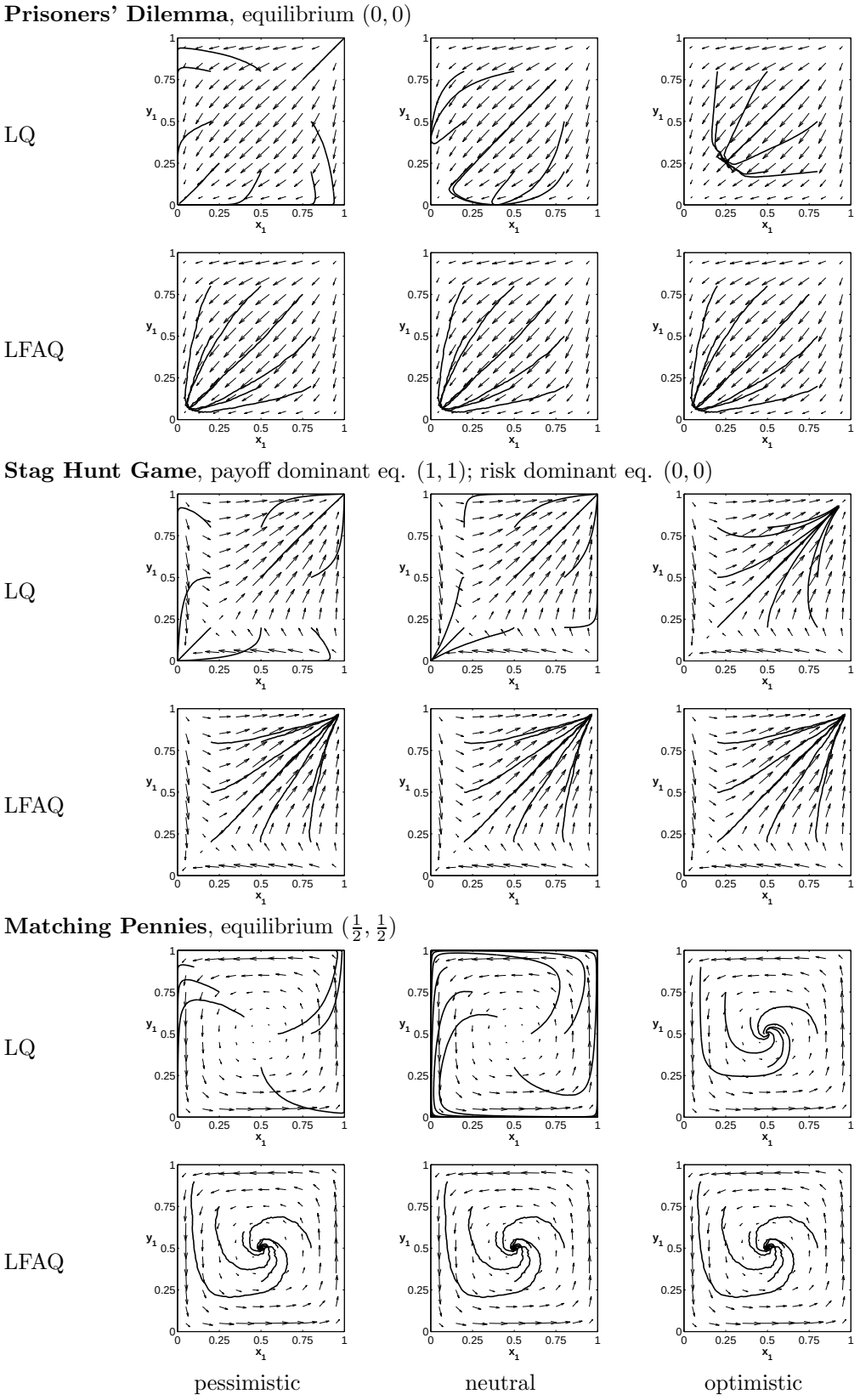


Figure 6.1: Overview of the behavior of Lenient Q-learning and Lenient FAQ-learning in different games. The figure shows different initialization settings for the Q values: pessimistic (left), neutral (center) and optimistic (right). The arrows represent the directional field of the evolutionary model derived by Panait *et al.* (2008).

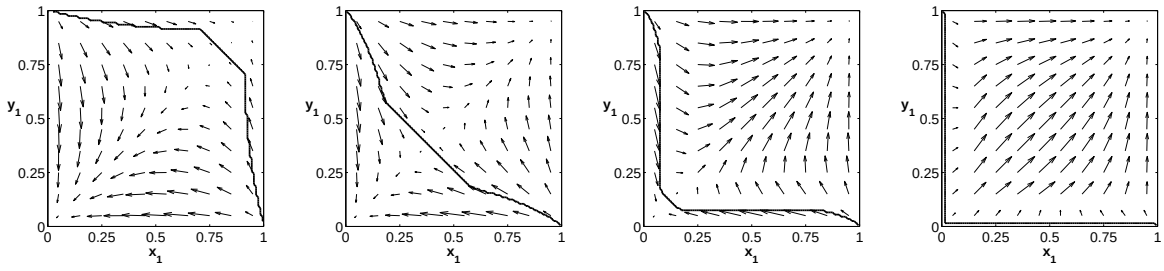
Figure 6.1 shows several policy trajectories of both LQ and LFAQ for the three different games and initialization settings. The directional field of the evolutionary model is depicted as well. Both learners use $\tau = 0.1$, $\gamma = 0.9$ and $\kappa = 5$. A learning rate of 10^{-5} is chosen in order to obtain predictable behavior; with the given value for κ this relates to $\alpha = 5 \cdot 10^{-5}$ for LQ, and $\alpha = 5 \cdot 10^{-2}$ and $\beta = 10^{-3}$ for LFAQ.

These results show that the behavior of LQ indeed depends on the initial Q-values. This leads to considerable differences in behavior and convergence properties of the learner. For example, the neutral and optimistic initialization in the Prisoners' Dilemma lead to non-convergence, at least not to the pure Nash equilibrium. The same holds for the pessimistic and neutral initialization in the Matching Pennies game, where the learning trajectories spiral outwards and away from the equilibrium. In the Stag Hunt game, LQ converges to either one of the equilibria depending on the initial settings. LFAQ on the other hand is more robust; its behavior is relatively independent of the initialization. In each of the examples it converges to the game's Nash equilibrium, and in the Stag Hunt game all learning trajectories indeed converge to the pay-off dominant Nash equilibrium, which is the Pareto optimum of the game.

Comparing the learners' trajectories to the evolutionary model, it is clear that LQ deviates from the expected path in most scenarios, whereas LFAQ shows behavior consistent with the predicted dynamics. The two learning algorithms behave most similar in the Stag Hunt and Matching Pennies game with optimistic initial Q-values. However, in many applications the rewards are not known in advance, which makes the selection of proper initial values infeasible. This makes LFAQ a better choice than LQ, as its behavior does not depend on the initialization and is consistent with the preferable dynamics of the evolutionary model derived by Panait *et al.* (2008).

6.1.2 Advantage of leniency in cooperative games

Leniency is a concept specifically suited to cooperative games. It was introduced to overcome the problem that initial mis-coordination may lead to suboptimal convergence (Panait *et al.*, 2006). This advantage can be illustrated by comparing the dynamics of Lenient FAQ for different degrees of leniency. Figure 6.2 shows the dynamics of LFAQ in the Stag Hunt game for various values of κ; the scenario where $\kappa = 1$ corresponds to the non-lenient FAQ algorithm. This example demonstrates that a higher degree of leniency leads to a larger basin of attraction for the Pareto optimal equilibrium of the game, which is at $(1, 1)$. In the limit, the basin of attraction consumes the whole strategy space, which indicates that the probability of converging to the optimal solution can be raised arbitrarily close to 1 by increasing the value of κ.



$\kappa = 1$ $\kappa = 2$ $\kappa = 5$ $\kappa = 25$

Figure 6.2: The dynamics of Lenient Frequency Adjusted Q-learning in the Stag Hunt game with varying degree of leniency. The line indicates the border between the two basins of attraction for the risk dominant equilibrium at $(0, 0)$ and the payoff dominant equilibrium at $(1, 1)$.

These results illustrate the claim of Panait *et al.* (2008) that "properly-set lenient learners are guaranteed to converge to the Pareto-optimal Nash equilibria in coordination games". Moreover, the previous section demonstrated that LFAQ indeed matches the evolutionary model on which this claim was based; therefore LFAQ inherits the theoretical guarantees from the evolutionary model.

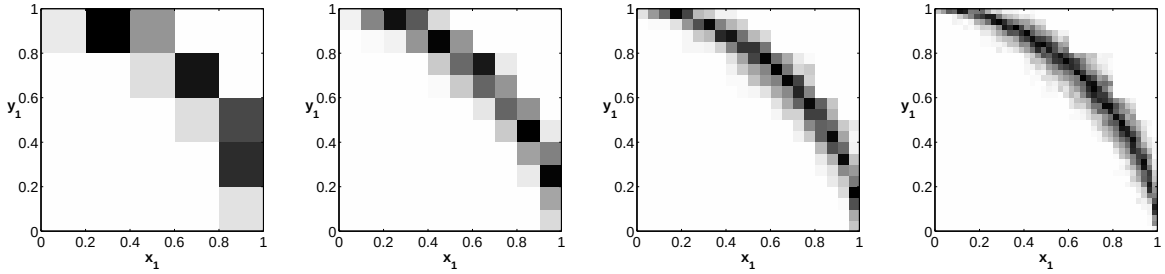
6.2 Fine-tuning the learning rate

The learning rate of the investigated RL algorithms needs to be fine tuned in two ways, as explained in Section 5.2.1. First of all, the right balance has to be struck between predictability of the learner on the one hand, and its speed of convergence on the other. Secondly, the relation between an algorithm's learning rate and its respective step size parameters needs to be investigated. Knowledge of this relation can be used to make sure that opposing learners converge comparably fast, thereby ensuring a fair competition. Experiments are conducted to find an answer to both problems; this section presents the results.

6.2.1 Predictability of the learning behavior

As discussed previously, a larger step size leads to a faster learning process but also to larger deviations in learning behavior. A large step size may even cause the learning process to become unstable. A very small step size on the other hand increases the time needed for the learner to converge, which is infeasible in practice.

In order to analyze the behavior of the different RL algorithms with varying step size, a gradient field is constructed that shows the convergence pureness over the policy space (see Section 5.4.2). This gradient field is defined as a grid over the policy space, where the convergence pureness, or certainty, is calculated for each cell by simulating the learning process several times, with uniformly distributed starting points within the grid cell, and investigating to which equilibrium the algorithm converges. In order to reduce the number of simulations needed, this gradient field is gradually refined by increasing the resolution only in those parts of the policy space that are still uncertain. Figure 6.3 shows how the resolution of a gradient field is step-wise increased. These gradient fields show the convergence pureness, averaged over 100 simulations for each grid cell, of FAQ-learning in the Stag Hunt game with $\alpha = \beta = 0.01$, $\tau = 0.01$ and $\gamma = 0$. White areas indicate complete certainty with respect to the equilibrium the learner will converge to, whereas darker areas indicate increasing uncertainty; black indicates a 50/50 chance of converging to either one of the pure equilibria.



5×5 grid 10×10 grid 20×20 grid 40×40 grid

Figure 6.3: Gradual refinement of convergence pureness gradient fields in the Stag Hunt game using FAQ-learning. White areas are certain, black areas are uncertain.

To illustrate the meaning of these results, Figure 6.4 shows the directional field of the evolutionary model for the same game and learning algorithm. The figure shows that the region of uncertainty indicated by the gradient field matches the border between the two basins of attraction in this game.

Figure 6.5 shows the convergence pureness gradient fields of FALA for different step sizes α in Battle of the Sexes. The convergence pureness is averaged over 100 simulations for each grid cell (10 for $\alpha = 0.001$). This example illustrates how the convergence pureness relates to the step size that is used in the learning algorithm. It is clear that a larger step size results in a larger area of uncertainty, where the learning process may converge to either equilibrium. When the step size decreases the area of uncertainty decreases. As can be seen, for $\alpha = 0.001$ only the border between the two basins of attraction is uncertain. This area consists almost entirely of grid cells that overlap the border, which will therefore always remain uncertain. However, the area can be made arbitrarily small by increasing the resolution.

The results of the convergence pureness experiments are summarized in Table 6.1. The table shows the percentage of the policy space for which convergence is certain for each of the learning algorithms in three different games. The choice of games is based on the fact that these all have more than one

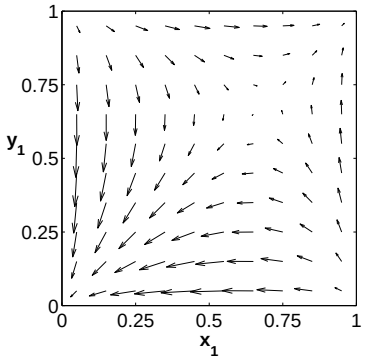
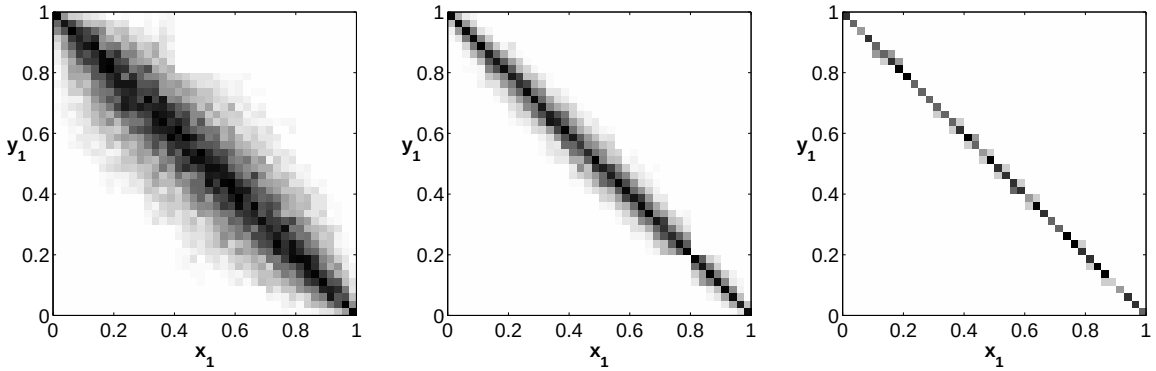


Figure 6.4: Replicator dynamics of FAQ learning in the Stag Hunt game.



$\alpha = 0.1$ $\alpha = 0.01$ $\alpha = 0.001$

Figure 6.5: Gradient field comparison of the convergence pureness of FALA in Battle of the Sexes using different learning rates. Grid resolution is 40×40.

equilibrium, which is required for the concept of convergence pureness to make sense. The table also shows the average percentage over these three games. Several step sizes are taken into account, indicated by the value of α in the table. Note that for notational convenience the step size parameter λ of RM is assumed to be equal to α in this case. The other parameters are $\beta = \tau = 0.01$ and $\gamma = 0$ for FAQ and LFAQ, and $\kappa = 5$ for LFAQ. Simulations are run for a maximum of $100 \cdot \alpha$ iterations ($500 \cdot \alpha$ for LFAQ) or until the learning process has ϵ-converged to an equilibrium with $\epsilon = 10^{-4}$ (see Section 5.4.2). In practice almost all simulations converged well before the maximum number of iterations was reached, the only exception being LFAQ in the Stag Hunt game, with $\alpha = 0.1$. In this case, 98.4% of all simulations converged.

The table (6.1) shows that the predictability of the learning process indeed increases with a decreasing step size. A step size of 0.1 in these games is clearly too large, since only little over 50% of the policy space is certain. With a step size of 0.01 the area of certainty increases to around 83%, and when the step size is 0.001 over 97% of the policy space is certain. Important to note is that by way of construction an overal certainty of 100% is infeasible, since there will always be grid cells overlapping the border between the basins of attraction. For example, in the Battle of the Sexes the border is on the diagonal, meaning that 40 cells will overlap the border when a resolution of 40×40 is used. As a result, the maximum percentage of certainty will be around $100 - \frac{40}{1600} \cdot 100 = 97.5\%$, which is similar to the results obtained using a learning rate α of 0.001.

Interesting to note is that the RM algorithm already shows a reasonable certainty overall with a large step size. LFAQ behaves more predictable than FAQ and FALA in the Stag Hunt and Coordination Game, where one equilibrium is preferred over the other. The gradient fields show that in these games the LFAQ learner indeed converges with higher certainty to the preferred equilibrium. This demonstrates again that leniency towards the other player in cooperative games results in a higher performance. The differences between the learning algorithms diminish when the learning rate decreases: when $\alpha = 0.01$

Table 6.1: Percentage of the policy space for which convergence is certain, based on the value of α, in the Prisoners' Dilemma (PD), Stag Hunt (SH), Battle of the Sexes (BoS) and Coordination Game (CG). Results are acquired using a grid resolution of 40×40.

$\alpha = 0.1$	SH	BoS	CG	Avg
FAQ	18.4	34.6	43.5	32.2
LFAQ	70.4	36.3	78.0	61.5
RM	80.8	72.3	77.8	77.0
FALA	31.6	45.0	49.1	41.9
Avg	50.3	47.0	62.1	53.1

$\alpha = 0.01$	SH	BoS	CG	Avg
FAQ	74.4	78.6	80.3	77.8
LFAQ	86.0	85.6	90.5	87.4
RM	91.9	90.9	91.3	91.4
FALA	72.4	79.3	80.3	77.4
Avg	81.2	83.6	85.6	83.5

$\alpha = 0.001$	SH	BoS	CG	Avg
FAQ	96.9	95.9	97.4	96.8
LFAQ	97.9	96.6	98.5	97.6
RM	98.6	97.1	98.6	98.1
FALA	95.4	96.3	97.1	96.3
AVG	97.2	96.5	97.9	97.2

LFAQ and RM still outperform the other algorithms, but only with a small margin; when $\alpha = 0.001$ the difference in certainty is further reduced to only $1 \sim 2\%$.

A different way to look at the behavior of the learning algorithms for various step sizes is to calculate the standard deviation over time of several learning trajectories, starting at a single point. In this case, the standard deviation is based on the Euclidean distance between the trajectory points $[x_1(t), y_1(t)]$ for different simulation runs (see Equation 5.8). Figure 6.6 shows the standard deviation over time for the different RL algorithms in the Prisoners' Dilemma. The settings for *alpha* are given, the other parameters of FAQ and LFAQ are set as follows: $\beta = 0.01$, $\tau = 0.01$, $\gamma = 0.9$ and $\kappa = 5$. The figure also shows the corresponding average learning trajectories. The average and standard deviation are calculated over 100 simulations (10 for $\alpha = 0.0001$), all starting at $[x_1 = 0.4, y_1 = 0.75]$.

These results confirm that the predictability of the learners' behavior increases with decreasing step size. Furthermore, where the convergence pureness experiments only investigate to which equilibrium the learners converge, the standard deviation plot also illustrates in what way they do so. The results indicate that even if the convergence is predictable, there can still be considerable variation in the learning trajectories that lead to the equilibrium. Figure 6.7 further highlights this variation by showing the individual learning trajectories together with their average for different step sizes.

The results presented in this section show that the predictability of a learning algorithm's behavior indeed increases with decreasing step size. Overall, the Regret Minimization algorithm behaves more predictable than the other learning algorithms, both with respect to convergence as to its learning trajectory, especially when the step size is large ($\alpha = 0.1$). For smaller step sizes the differences between the RL algorithms diminish. Both Table 6.1 and Figure 6.6 indicate that a step size α of 0.1 results in unpredictable behavior for FAQ, LFAQ and FALA, whereas RM still performs reasonably well. Furthermore, no considerable differences are observed in the behavior of those three algorithms, except that LFAQ tends to perform better in cooperative games where one equilibrium is preferred over the other.

To conclude, in order to achieve reliable results in the remainder of the experiments in this thesis, the step size for FAQ, LFAQ and FALA has to be at most 0.01. For RM, a step size of 0.1 is also allowed. The next section investigates how the various step sizes relate to the algorithms' speed of convergence.

6.2.2 Convergence speed

The next step in fine-tuning the learning rate is to investigate the convergence speed of the learning algorithms for various step sizes. This is important in two ways. First of all, it gives an idea about the relation between the step size and the number of iterations needed to converge, which helps to find the right balance between predictability and speed of the learning process. Secondly, it shows the speed of the learning algorithms relative to each other. This is particularly interesting when two different learning algorithms are plays against each other. For example, knowing the relation between the learners' step sizes and their convergence speed makes it possible to select the step sizes in such a way that the different

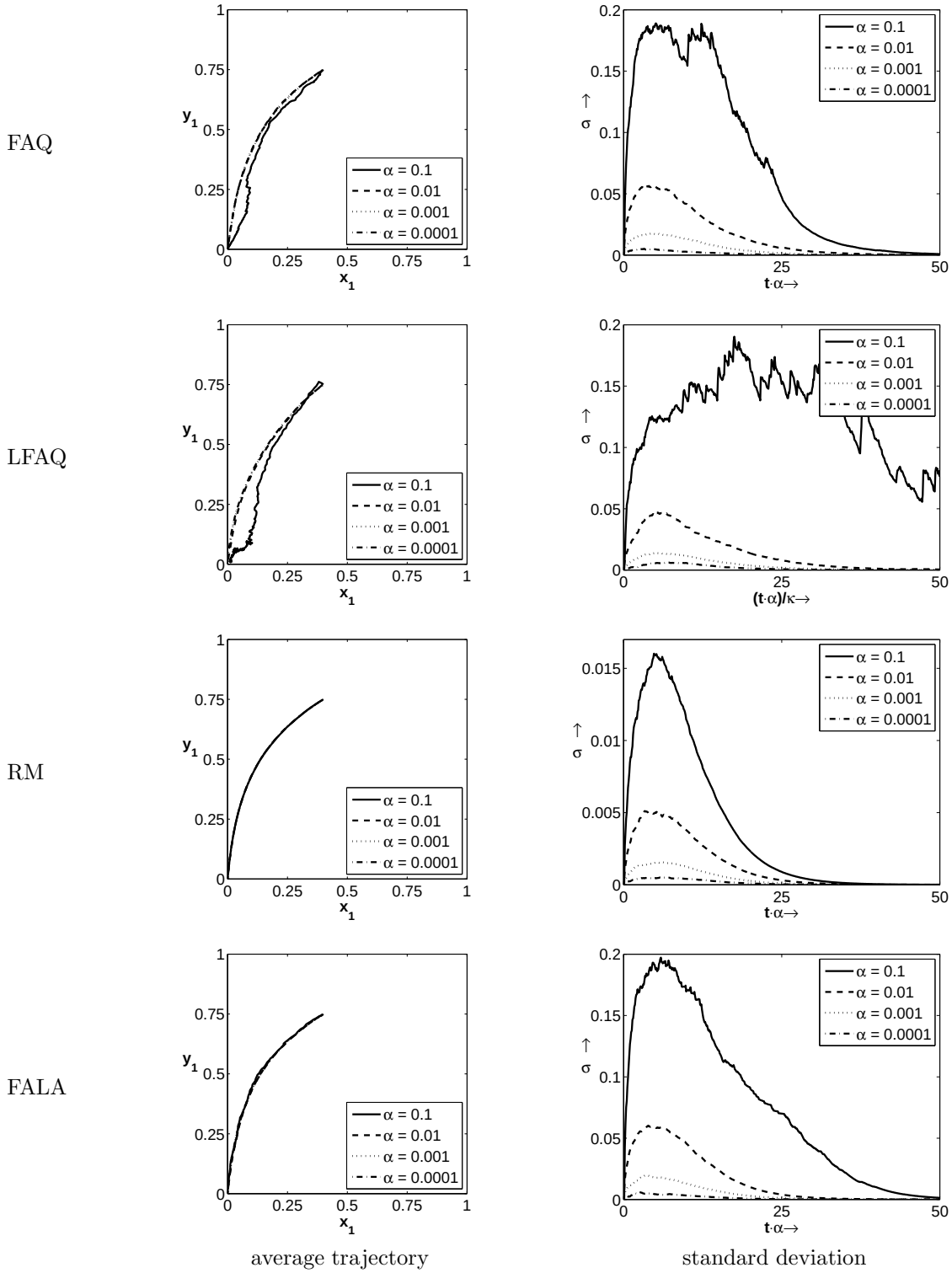
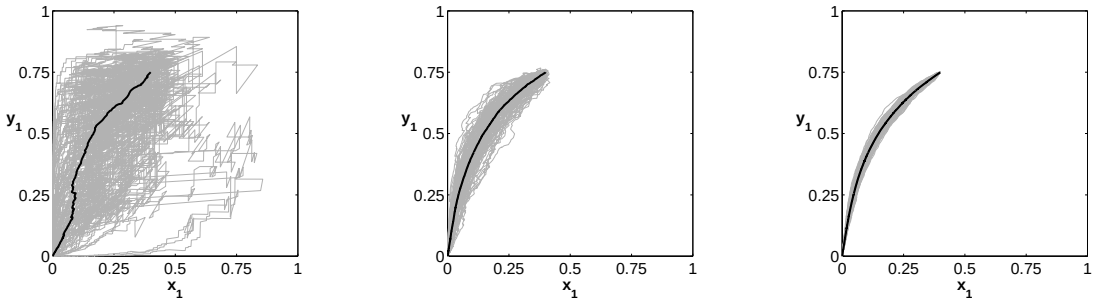


Figure 6.6: The average trajectory (left) and standard deviation over time (right) for varying learning rates in the Prisoners' Dilemma. All learners start at $[0.4, 0.75]$.



$\alpha = 0.1$ $\alpha = 0.01$ $\alpha = 0.001$

Figure 6.7: Individual FAQ-learning trajectories (gray) and their mean (black) in the Prisoners' Dilemma for different values of α, starting at $[0.4, 0.75]$.

algorithms learn comparably fast in self play, which ensures a fair competition in mixed play.

Table 6.2 shows the average number of iterations needed to converge for the different learning algorithms, in different games using varying settings for the step size α. These averages are calculated by running 2500 simulations (250 for $\alpha = 0.001$) with starting points uniformly distributed over the policy space. The same parameter settings are used as for the gradient field experiments in the previous section: $\beta = \tau = 0.01$ and $\gamma = 0$ for FAQ and LFAQ, and $\kappa = 5$ for LFAQ.

Table 6.2: Mean convergence time, rounded to integers, for varying step size α in the Prisoners' Dilemma (PD), Stag Hunt (SH), Battle of the Sexes (BoS) and Coordination Game (CG). Averaged over 2500 simulations with uniformly distributed starting points (250 simulations for $\alpha = 0.001$).

PD	$\alpha = 0.1$	$\alpha = 0.01$	$\alpha = 0.001$
FAQ	485	4667	46388
LFAQ	2907	26930	270003
RM	386	3802	38413
FALA	399	3828	38116

BoS	$\alpha = 0.1$	$\alpha = 0.01$	$\alpha = 0.001$
FAQ	169	1621	16280
LFAQ	1296	11341	113151
RM	155	1485	14827
FALA	149	1473	14898

SH	$\alpha = 0.1$	$\alpha = 0.01$	$\alpha = 0.001$
FAQ	171	1666	16163
LFAQ	2575	13491	121650
RM	166	1586	15778
FALA	135	1497	15621

CG	$\alpha = 0.1$	$\alpha = 0.01$	$\alpha = 0.001$
FAQ	118	1116	11131
LFAQ	642	6017	60070
RM	112	1065	10611
FALA	111	1058	10576

The table indicates a clear relation between the step size α and the convergence speed: when the step size decreases with a factor 10, the number of iterations needed increases also with a factor 10. This relation becomes even more apparent when the results are plotted on a log-log scale, as shown in Figure 6.8 for FAQ learning. The only exception to this rule is LFAQ with $\alpha = 0.1$ in the Stag Hunt game. In this case, the relatively large step size apparently causes the learner to behave differently and as a result it converges more slowly. An analysis of the individual learning trajectories also indicates this difference. However, an in-depth investigation of this deviation falls outside the scope of this thesis, and it will therefore be treated as an outlier.

These results also provide a means to ensure comparable convergence speeds for the different algorithms. This is of particular importance for the mixed play experiments, in order to allow for a fair competition that emphasizes the true qualitative differences between the learners. For example, to match the convergence speed of FAQ and RM, the ratio ρ between their respective average number of iterations k, normalized for α, can be calculated as

$$\rho_{RM,FAQ} = \frac{\alpha \cdot k_{RM}}{\alpha \cdot k_{FAQ}}.$$

This ratio might differ depending on the game considered. In the Prisoners' Dilemma the mean ratio

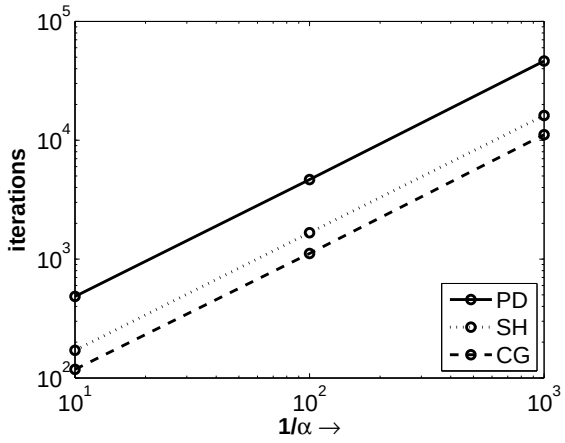


Figure 6.8: Convergence speed against step size α for FAQ learning on a log-log scale.

between RM and FAQ is

$$\frac{38.6 + 38.02 + 38.413}{48.5 + 46.67 + 46.388} = 0.81.$$

Since the relation between step size and convergence speed is inverse linear, multiplying the step size α of RM by ρ should alleviate the difference in convergence speed of FAQ and RM. Table 6.3 shows the resulting mean convergence time after applying this procedure for the different learning algorithms in the Prisoners' Dilemma. The ratio is calculated with respect to FAQ for all learners.

Table 6.3: The leveling effect of the ratio ρ on the mean convergence time of the different learners in the Prisoners' Dilemma. Averaged over 2500 simulations with uniformly distributed starting points (250 simulations for $\alpha = 0.001\rho$).

	ρ	$\alpha = 0.1\rho$	$\alpha = 0.01\rho$	$\alpha = 0.001\rho$
FAQ	1.00	485	4667	46388
LFAQ	5.86	532	5145	46625
RM	0.81	474	4686	46655
FALA	0.82	477	4648	46121

As can be seen, applying the ratio ρ to the step size of the learners indeed removes the initial differences in convergence time. The leveling effect is most clear for $\alpha = 0.001$; for larger step sizes some variation in the results can be observed. The relatively large deviations in the LFAQ results are caused by the fact in this case $\rho > 1$, which means that the step size further increases, leading to less predictable behavior. Applying this procedure to the other games, using the corresponding values for ρ, leads to similar results. Table 6.4 summarizes the ratios for all games and learners, calculated using Table 6.2. The values found in Table 6.4 are used throughout the remainder of the experiments.

Table 6.4: The ratio ρ for different games and learners, with respect to FAQ.

	PD	SH	BoS	CG
FAQ	1.00	1.00	1.00	1.00
LFAQ	5.86	7.81	7.20	5.41
RM	0.81	0.97	0.91	0.95
FALA	0.82	0.89	0.90	0.95

As concluded in Section 6.2.1, the step size of the learning algorithms should be at most 0.01 (or 0.1 for RM). Setting $\alpha = 0.001$ as base step size parameter, in combination with the ratios found in Table 6.2, leads to a maximum step size of $0.001 \times 7.81 = 0.00781$ for LFAQ which is just below the maximum of 0.01. Therefore, $\alpha = 0.001$ is selected as base parameter setting for the homogeneous and heterogeneous experiments described in the remainder of the chapter.

6.3 Self play

Self play, or homogeneous play, is the standard form of learning experiments, in which each competing player implements the same learning algorithm. All experiments described so far fall under this heading. Self play experiments can provide useful insights into the standard behavior of the respective learning algorithms, and therefore provide a baseline for the heterogeneous mixed play experiments in Section 6.4.

The same parameter settings are used in all experiments described in this section. All algorithms use step size $\alpha = \lambda = 0.001$ times the ratio given in Table 6.4. For (L)FAQ, $\beta = 0.01$, $\tau = 0.01$ and $\gamma = 0$; LFAQ uses $\kappa = 5$; and FALA uses the L_{R-I} update scheme where $\beta = 0$. This section is further divided in three parts, describing the behavior, convergence properties, and performance of the learning algorithms respectively.

6.3.1 Behavior

The behavior of a learner over time can be visualized using a trajectory plot or by plotting the directional field of the corresponding evolutionary model (see Section 5.4.1). Here, a combination of both is used to show how the individual learning trajectories relate to their evolutionary predication. All trajectory plots show the average trajectory over 10 simulations of 50,000 iterations each (100,000 for the Prisoners' Dilemma).

In the Prisoners' Dilemma, belonging to the first category of games, not much variation is observed in the trajectories or predictions for the different learning algorithms, they all behave similarly. Figure 6.9 shows the behavior of the four different algorithms in this game. As can be seen, all trajectories converge to the game's Nash equilibrium (D,D), which in the plot lies at (0,0). The directional field indicates that indeed all possible initial policies will eventually converge to this equilibrium.

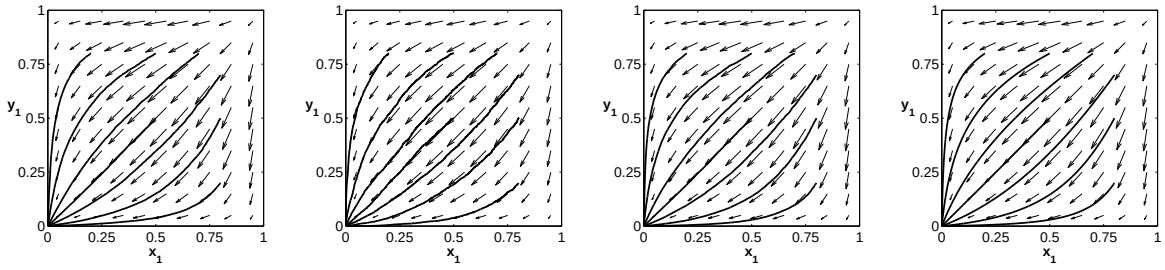


Figure 6.9: Several policy trajectories of the four different algorithms in the Prisoners' Dilemma. The arrows indicate the directional field of the corresponding evolutionary models.

In the three games belonging to the second category, with multiple equilibria, a clear distinction can be observed between the three non-lenient learners, FAQ, FALA and RM, and the lenient learner LFAQ. Figure 6.10 shows the difference in behavior of FAQ and LFAQ in these three games. In this case, FAQ is taken as a representative example of the non-lenient learners, whose behavior is again very similar to each other.

In the Battle of the Sexes, the trajectories converge to the same equilibria for both types of learners, but do so in different ways. Where the dynamics of the non-lenient learners correspond to the basic replicator dynamics of the game (see Figure 3.7), the lenient learner has a different mixed equilibrium much closer to (O,F) which indicates that a lenient player sticks to its preferred action much longer by ignoring lower payoffs. The higher the degree of leniency, the closer the mixed equilibrium gets to (O,F).

The Stag Hunt, and to a lesser extend also the Coordination Game, clearly show the advantage of leniency in cooperative environments. In these games, both player prefer the same equilibrium, which is in both cases located at (1,1). In the Stag Hunt, non lenient learners prefer the safer risk dominant equilibrium (0,0), where both hunt for Hare. The lenient learner in this case ignores initial mis-coordinations, and can therefore reach the Pareto optimal equilibrium (S,S) in most cases. In the Coordination Game this effect is less strong, since neither equilibrium presents a safe choice. However, again the lenient learner is able to reach the Pareto optimal equilibrium more often. This result is analyzed in more detail in the next section.

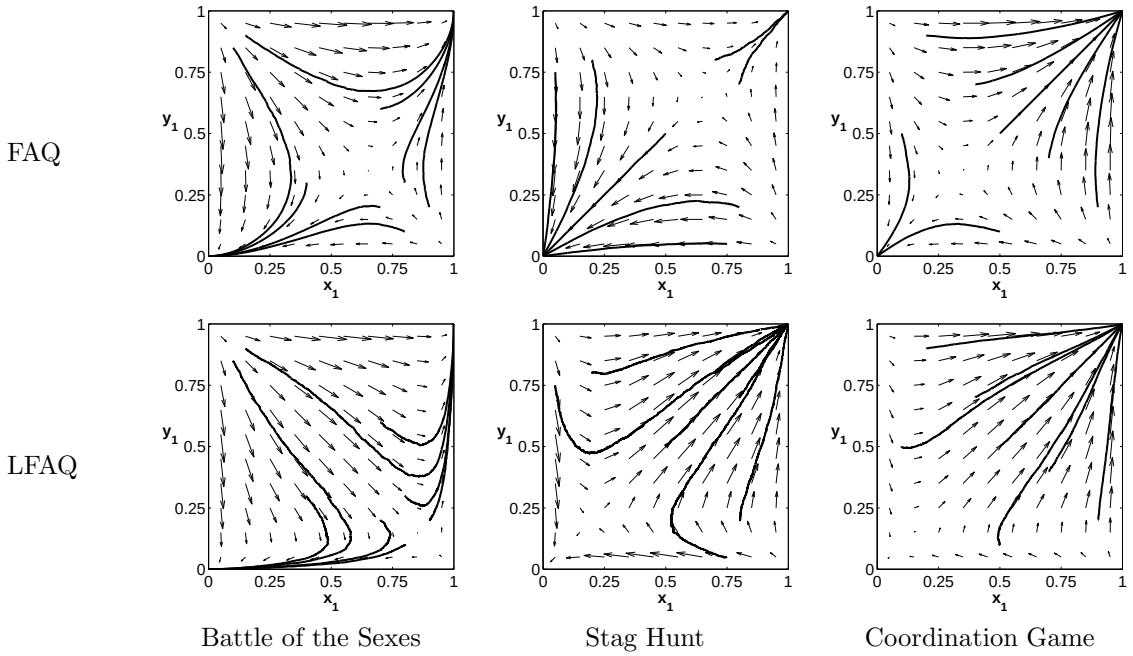


Figure 6.10: Policy trajectories of FAQ and LFAQ in the Battle of the Sexes (BoS), Stag Hunt (SH), and Coordination Game (CG). In these games, the trajectories of FALA and RM are very similar to those of FAQ.

The last game, Matching Pennies, shows different results for each learner. The policy trajectories are presented in Figure 6.11. In this case, both FAQ and LFAQ are spiraling inward towards the equilibrium; LFAQ spirals faster but moves slower. The other two learners, especially the RM algorithm, lead to concentric policy trajectories around the equilibrium. This behavior corresponds to the basic replicator dynamics of the game shown in Figure 5.6.

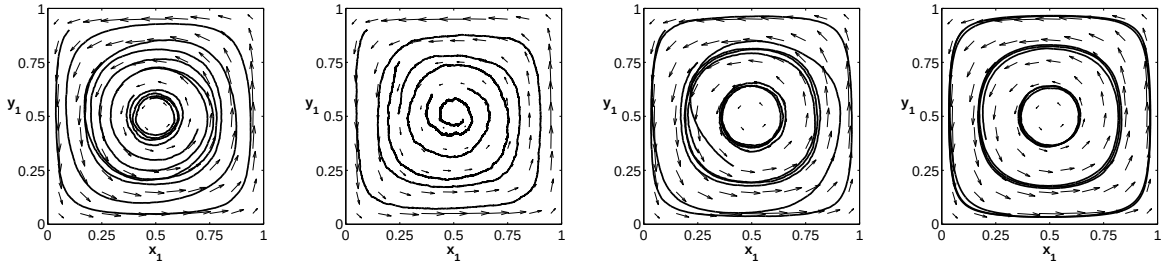


Figure 6.11: Policy trajectories of the four different learning algorithms in the Matching Pennies game.

Summarizing, it can be argued that in general the behavior of LFAQ deviates most from the other learners. This deviation occurs most notably in games with multiple equilibria, where cooperation becomes important. In these cases, the lenient learner is able to reach to Pareto optimal Nash equilibrium more often than the three non-lenient learners.

6.3.2 Convergence properties

As mentioned in the previous section, in cooperative games, with multiple equilibria, a player might prefer one equilibrium over the others. In the Stag Hunt, both players are best off in the Pareto optimal equilibrium (S,S), however they might still prefer (H,H) for its safety. In the Battle of the Sexes, both players prefer opposite equilibria. Finally, in the Coordination Game both players prefer the Pareto optimal equilibrium (O,O).

A useful concept to describe these preferences is the basin of attraction (see Section 3.2.3). The basin of attraction of an equilibrium can be found by calculating the area of the policy space for which the learning process eventually converges to that equilibrium, as explained in Section 5.4.2. Table 6.5 shows the percentage of the policy space belonging to the basin of attraction of the two Nash equilibria in the three cooperative games. These percentages are calculated using a grid resolution of 100×100. The other two games are not shown since they have only one equilibrium and therefore only one basin of attraction, which is either the whole policy space - in the Prisoners' Dilemma, and in Matching Pennies when the learning trajectories are spiraling inwards - or empty - in Matching Pennies when the learning trajectories form concentric circles.

Table 6.5: Percentage of the policy space belonging to the basin of attraction of the stable Nash equilibria in cooperative games, for different learners. Pareto optimal equilibria are indicated with $*$.

	SH		BoS		CG	
	(H,H)	(S,S)*	(F,F)	(O,O)	(F,F)	(O,O)*
FAQ	74.3	25.7	49.5	49.5	25.7	73.9
LFAQ	19.0	80.9	49.5	49.5	10.8	89.2
FALA	73.8	26.3	49.5	49.5	26.3	73.8
RM	73.8	26.3	49.5	49.5	26.3	73.8

These results again indicate that LFAQ has a higher probability of reaching the Pareto optimal equilibria in the Stag Hunt and the Coordination Game. Especially in the Stag Hunt, LFAQ outperforms the other learners by a considerable margin of 80% against 26%. In the Battle of the Sexes both equilibria are Pareto optimal, and neither is preferred by both players. As a result, the policy space is divided equally among those two equilibria. Note that in some cases a small percentage of the policy space is not assigned to a basin of attraction. This may happen when grid cells are overlapping the border between two basins of attraction. In this case, the calculation method might not be able to assign this cell to either of the equilibria.

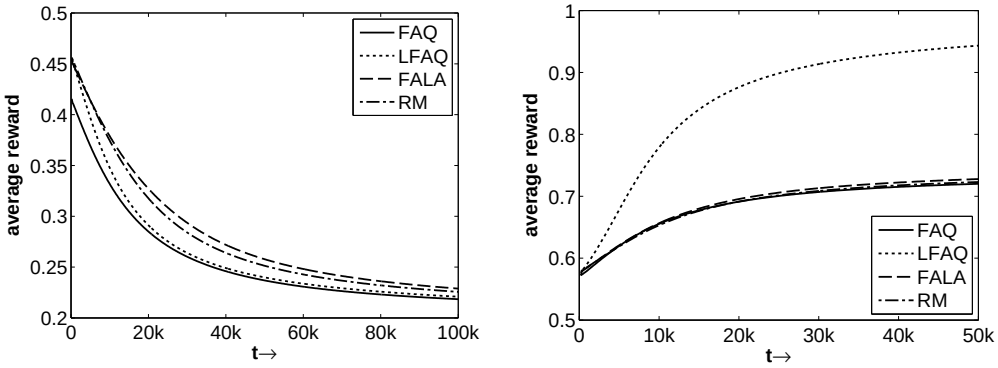
6.3.3 Performance

The performance of the learners is analyzed by looking at their average cumulative reward. Table 6.6 shows the cumulative reward of the learning algorithms in four different games. The numbers are acquired by running 1,000 simulations with uniformly distributed starting points for each combination of game and learner, and averaging over the result of both players to rule out deviations based on player type rather than learning algorithm. The cumulative rewards for playing the games' Nash equilibria are also given to illustrate the relative performance of the learning algorithms. For the Battle of the Sexes only one value is given as both equilibria are equally likely in this case; for the Stag Hunt and the Coordination Game values are given for both pure equilibria. The Matching Pennies game is not show in this table since this is a zero-sum game (be it normalized), and therefore the average reward over the two players is constant, in this case $\frac{1}{2}$ per iteration.

Table 6.6: The average cumulative reward, rounded to integers, of the learning algorithms in different games, after 50,000 iterations (100,000 iterations for the Prisoners' Dilemma). The cumulative rewards for playing the games' Nash equilibria are given for comparison.

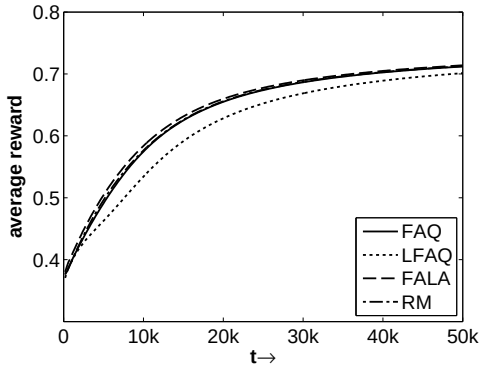
Performance	PD	SH	BoS	CG
FAQ	21839	36010	35608	43958
LFAQ	22084	**47173**	35075	**48054**
FALA	**22880**	36394	**35703**	42347
RM	22558	36155	35629	41205
Nash	20000	33333, 50000	37500	25000, 50000

The results show that there is not much difference in performance in the Prisoners' Dilemma and the Battle of the Sexes. In these two games, the basins of attraction are similar for all learners, as explained in the previous section. Furthermore, the differences in behavior of the learners are not significant enough to result in large deviations in cumulative reward. In the Stag Hunt and Coordination Game, however,

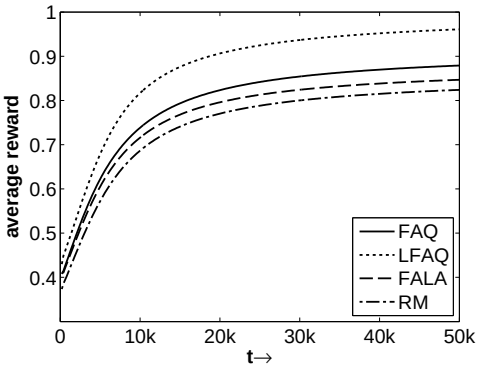


Prisoners' Dilemma

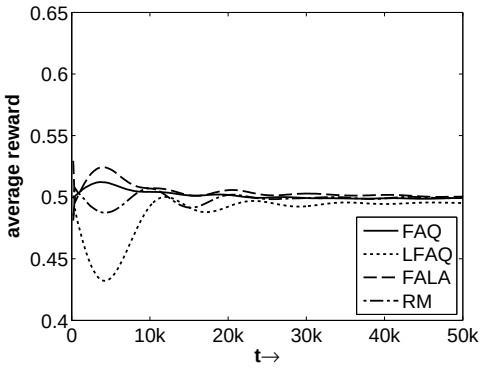
Stag Hunt



Battle of the Sexes



Coordination Game



Matching Pennies

Figure 6.12: Average reward over time for different learners. The values are averaged of both players, except in Matching Pennies where only the reward of player 1 is taken into account.

a difference can be observed between LFAQ and the other, non-lenient, learners. In these cases, LFAQ has a higher probability of reaching the Pareto optimal equilibrium than the other learners, leading to a higher average cumulative reward. In both games the average cumulative reward of LFAQ is close to the reward for playing the Pareto optimal Nash equilibrium.

The performance of a learner can be studied in more detail by looking at the development of the average reward over time. Figure 6.12 compares the average reward over time of different learners in each game. For Matching Pennies only the reward of player 1 is taken into account, for the other games the reward is again averaged over both players.

These figures tell the same story: LFAQ outperforms the other learners in the Stag Hunt and the Coordination Game, whereas in the other games there is less deviation in the results. Furthermore, two

interesting details can be noted. First, in the Prisoners' Dilemma LFAQ converges more quickly in the beginning, leading to a steeper decrease in average reward initially. The reason is that the lenient learner ignores the relatively low reward of the Nash equilibrium and focuses on the maximum reward that results if the opponent defects. Secondly, in the Battle of the Sexes the reward of LFAQ increases less quick than that of the other learners, which is caused by the fact that LFAQ sticks to its preferred action for a longer time before conceding to its opponent, as explained earlier.

6.4 Mixed play

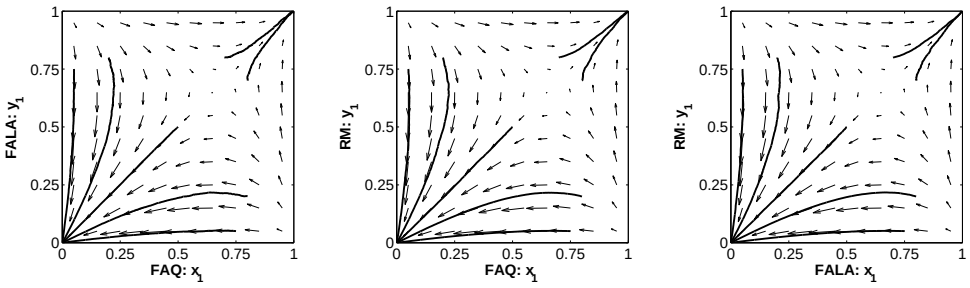
In the mixed play experiments, games are played by heterogeneous pairs of players, meaning that both players implement different learning algorithms. The results of these experiments are compared with those of the self play experiments in the previous section. This gives in insight into how the behavior of a learner depends on the behavior of its opponent. Moreover, the results indicate how well the learners fare against different opponents by looking at their performance.

Again, all experiments use the same parameter settings. All learners use step size $\alpha = \lambda = 0.001$ times the ratio given in Table 6.4. For (L)FAQ, $\beta = 0.01$, $\tau = 0.01$ and $\gamma = 0$; LFAQ uses $\kappa = 5$; and FALA uses the L_{R-I} update scheme where $\beta = 0$. This section is divided in three parts, describing the behavior, convergence properties, and performance of the learning algorithms respectively.

6.4.1 Behavior

The behavior of the learners is again analyzed by running simulations with several different starting points, and plotting the resulting trajectories together with the directional field of the mixed replicator equations of the evolutionary models. For each starting point, 10 simulations are run and the resulting average trajectories are shown. Each simulation consists of 50,000 iterations (100,000 in the Prisoners' Dilemma).

As in self play, the non-lenient learners behave very similar to each other; most deviations occur when LFAQ is involved. To illustrate, Figure 6.13 shows the behavior of combinations of the three non-lenient learners in the Stag Hunt game. The behavior of these combinations of learners does not show any significant differences. Moreover, the behavior is very similar to the self play behavior of these learners, see for example the self play of FAQ in Figure 6.10.



FAQ - FALA FAQ - RM FALA - RM

Figure 6.13: Policy trajectories of combinations of non-lenient learners in the Stag Hunt game.

When LFAQ is involved, the resulting behavior tends to differ from any of the learners' self play behavior. Figure 6.14 shows the behavior of a combination of FAQ and LFAQ in four different games. In these games, both FALA and RM in combination with LFAQ show very similar results. The learning trajectories clearly deviate from the self play behavior of any of the learners. Interesting to note is that the evolutionary prediction is still correct, which shows that also in a heterogeneous setting the evolutionary game theoretic approach provides useful insights.

In the Prisoners' Dilemma, the general behavior of the combination FAQ - LFAQ is still very similar to self play. The trajectories are lightly bend downwards, indicating that LFAQ converges slightly faster in the beginning of the learning process. This can be explained by the fact that the lenient learner ignores several lower rewards, and instead focuses on the game's maximum payoff that results when LFAQ plays

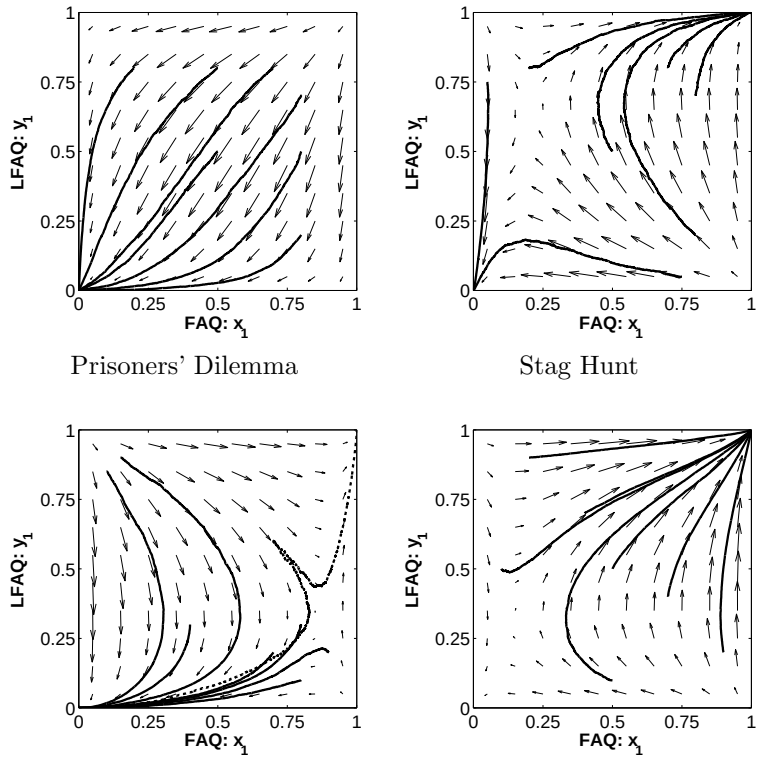


Figure 6.14: Policy trajectories of mixed play between FAQ and LFAQ in four different games. The dotted trajectory line in Battle of the Sexes indicates that not all simulations converged to the same equilibrium for this starting point; instead the average trajectory to each equilibrium is shown.

Defect and its opponent plays Cooperate. This drives the lenient learner towards action D early in the process.

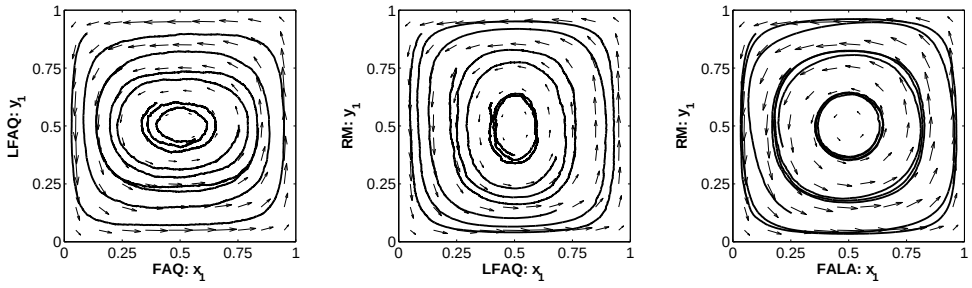
The examples of the Stag Hunt and the Coordination Game show again the advantage of leniency in a cooperative environment. The combination of LFAQ and FAQ converges to the Pareto optimal equilibrium more often than FAQ in self play. The strong inward curved trajectories in the Stag Hunt indicate that initially the learners struggle to settle on an equilibrium. LFAQ prefers the Pareto optimal equilibrium at (1,1), whereas FAQ prefers the risk dominant equilibrium at (0,0). Being lenient, LFAQ ignores the initial preference of FAQ to a certain degree, and is therefore able to steer the learning process towards the Pareto optimal solution.

The Battle of the Sexes proves interesting in a slightly different way. Here, in self play the learners behave similar in general, as neither equilibrium is preferred by both players. When combined, however, the otherwise symmetric learning dynamics are skewed in favour of LFAQ, resulting in a larger basin of attraction for its preferred equilibrium (F,F), at (0,0). This shows that leniency can also offer an advantage in coordination games with conflicting interests, when playing against non-lenient players.

Finally, in the Matching Pennies game, different behavior can again be observed for each combination of learners. Most notably, FAQ and LFAQ still tend to spiral inwards towards the mixed equilibrium, but the inwards movement decreases when either learner plays against FALA or RM. A combination of the latter two learners shows hardly any inward movement, in line with these learners' respective self play results. Figure 6.15 provides examples of each of these situations.

6.4.2 Convergence properties

The effect of the behavioral changes resulting from mixed play, described in the previous section, can best be analyzed by looking at the changing basins of attraction of the different games. As in self play, this analysis is limited to the three games with multiple equilibria, since both the Prisoners' Dilemma

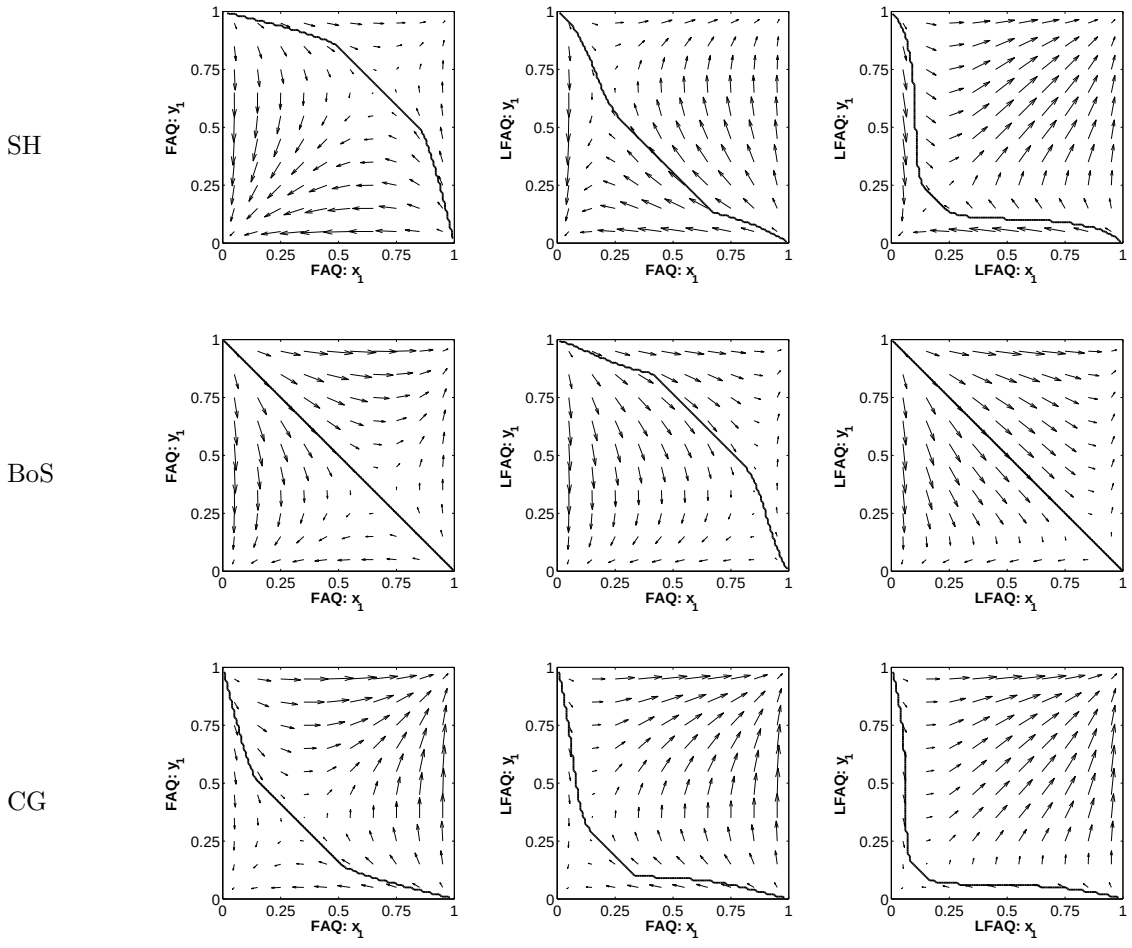


FAQ - LFAQ LFAQ - FALA LFAQ - RM

Figure 6.15: Policy trajectories of combinations of learners in the Matching Pennies game.

and the Matching Pennies game have only one basin of attraction that either fills the whole policy space, or is empty.

As noted before, the most interesting changes occur when LFAQ is involved. Figure 6.16 shows the basins of attraction in the three games for FAQ and LFAQ, both in self play and when the two play against each other. The basins are calculated using a grid resolution of 100×100, and are represented as a solid line indicating their border. The directional field of the (mixed) evolutionary model is also shown to provide a clear overview of the convergence properties.



FAQ self play FAQ vs. LFAQ LFAQ self play

Figure 6.16: Overview of the basins of attraction of FAQ, LFAQ, and the combination between both learners as representative examples of the difference between non-lenient and lenient learners.

The figures show several interesting properties of mixed play learning. In the Stag Hunt game, for example, both learners have almost oppositely distributed basins of attraction in self play, with FAQ converging to the risk dominant equilibrium at (0,0), and LFAQ to the payoff dominant equilibrium at (1,1) in the larger part of the policy space. When these two learners play against each other, the resulting basins of attraction appear to be a mix between those two opposites. A similar effect can be seen in the Coordination Game, although in this case the difference is much smaller as the original basins of attraction are more similar.

Also interesting to note is that in the Battle of the Sexes, FAQ and LFAQ show similar convergence properties in self play, but LFAQ profits in the mixed scenario: a larger part of the policy space converges to (0,0), which corresponds to the preferred equilibrium (F,F) of LFAQ, being player 2. When the learners switch sides, again LFAQ 'wins' and FAQ 'loses' in a larger part of the policy space.

The results for all combinations of learners are summarized in Table 6.7. The baseline values indicate the basins of attraction in self play, where 'baseline other' is the average of the three non-lenient learners as given in Table 6.5. The results are similar to the examples given above. In general, mixed play between non-lenient learners does not lead to significantly different result from the self play case. In common interest cooperative games, the results for mixed play involving LFAQ lie in between those of the self play scenarios. In cooperative games with conflicting interests, in this case the Battle of the Sexes, LFAQ 'wins' in a larger area of the policy space.

Table 6.7: Percentage of the policy space belonging to the basin of attraction of the stable Nash equilibria in cooperative games, for different combinations of learners. Pareto optimal equilibria are indicated with $*$.

	SH		BoS		CG	
	(H,H)	(S,S)*	(F,F)	(O,O)	(F,F)	(O,O)*
Baseline LFAQ	19.0	80.9	49.5	49.5	10.8	89.2
Baseline other	73.9	26.1	49.5	49.5	26.1	73.8
FAQ - LFAQ	37.3	62.7	68.3	31.7	16.7	83.3
FAQ - FALA	73.8	25.8	50.5	49.5	26.2	73.8
FAQ - RM	73.8	25.8	50.5	49.5	26.2	73.8
LFAQ - FALA	37.9	62.1	31.6	68.4	16.9	83.1
LFAQ - RM	37.9	62.1	31.2	68.8	16.9	83.1
FALA - RM	73.8	26.3	49.5	49.5	26.2	73.8

6.4.3 Performance

The performance of the learners is again analyzed by looking at the cumulative reward earned during game play. The cumulative reward of learning algorithm A against learning algorithm B is calculated as the average over 1,000 simulations where A is player 1 and B is player 2, and another 1,000 simulations where B is player 1 and A is player 2. This rules out deviations based on player type rather than learning algorithm. The starting points of the simulations are uniformly distributed over the policy space. The results are compared to those of the self play experiments in Section 6.3. Table 6.8 presents an overview of this comparison. The self play results are given in the diagonal for each game; the cumulative reward of learner A against learner B is given in row A and column B. The results for Matching Pennies are not shown in this table; in this game the results show almost no variation with an average of 25,000 and a standard deviation of 89.

In the Prisoners' Dilemma, the results do not show large differences between the learning algorithms. Apparently, the fact that LFAQ profits slightly in the beginning of the learning process does not result in a noticeably higher payoff. In the three cooperative games, however, LFAQ performs best against all other learners. Most notably, in Battle of the Sexes LFAQ clearly 'wins', as the other learners do worse against LFAQ than against themselves. Moreover, LFAQ does better against the others than against itself in this game. In the two common interest games, the Stag Hunt and the Coordination Game, LFAQ does worse against the others than against itself; in this case the other learners profit most in mixed play. Overall, however, it is clear that leniency is profitable in all coordination games, while at the same time is does not do worse in the non-cooperative games considered here.

Table 6.8: Overview of the performance (cumulative reward) of the learners (rows) against different opponents (columns).

PD	FAQ	LFAQ	FALA	RM
FAQ	21839	**22418**	22572	23237
LFAQ	22182	22084	**24281**	22422
FALA	21853	21451	22880	**23608**
RM	**22621**	21936	21740	22558

BoS	FAQ	LFAQ	FALA	RM
FAQ	35608	29170	34759	32617
LFAQ	**42020**	**35075**	**37825**	**40567**
FALA	36381	32842	35703	33945
RM	38463	30097	36871	35629

SH	FAQ	LFAQ	FALA	RM
FAQ	36010	43336	36295	37180
LFAQ	**42200**	**47173**	**42239**	**41223**
FALA	36205	42878	36394	36574
RM	37134	42232	36276	36155

CG	FAQ	LFAQ	FALA	RM
FAQ	43958	45191	44313	42937
LFAQ	**45191**	**48054**	**45228**	**45097**
FALA	44313	45228	42347	44546
RM	42937	45097	44546	41205

Figure 6.17 takes a closer look at the performance of lenient and non-lenient learners, showing the average reward over time for FAQ and LFAQ, both in self play and mixed play. In the Prisoners' Dilemma and the Matching Pennies game, not much variation is seen in the results, indicating that both learners do equally well in these games. In the Stag Hunt, LFAQ performs better than FAQ in self play, but it does worse in mixed play. This can be explained by the fact that FAQ still prefers to play action H in the beginning, which leads to a lower payoff for LFAQ when playing S. In the Battle of the Sexes, LFAQ clearly gains from mixed play, whereas FAQ looses. Finally, in the Coordination Game the mixed result lies between two two learner's results in self play. In this game, both players always receive the same payoff and therefore their cumulative reward in mixed play is exactly equal.

It is also possible to calculate the expected average reward of the learners, using the basins of attraction calculated in Table 6.7 and the games' payoff matrices. For example, self play of FAQ in the Stag Hunt game results in 74.3% of the policy space converging to (H,H) and 25.7% converging to (S,S). Given the game's normalized payoff matrix (Table 5.7), this leads to an expected average reward for FAQ of $0.743 \times \frac{2}{3} + 0.257 \times 1 = 0.75$. This expectation agrees with the trend observed in Figure 6.17. Similarly, the expected average rewards for LFAQ and a combination of both can be calculated, leading to the numbers shown in Table 6.9. Again, these evolutionary expectations are in line with the simulation-based findings presented in Figure 6.17, which shows that the replicator dynamics are not only useful in describing the behavior and convergence of the learners, but can also accurately predict their performance.

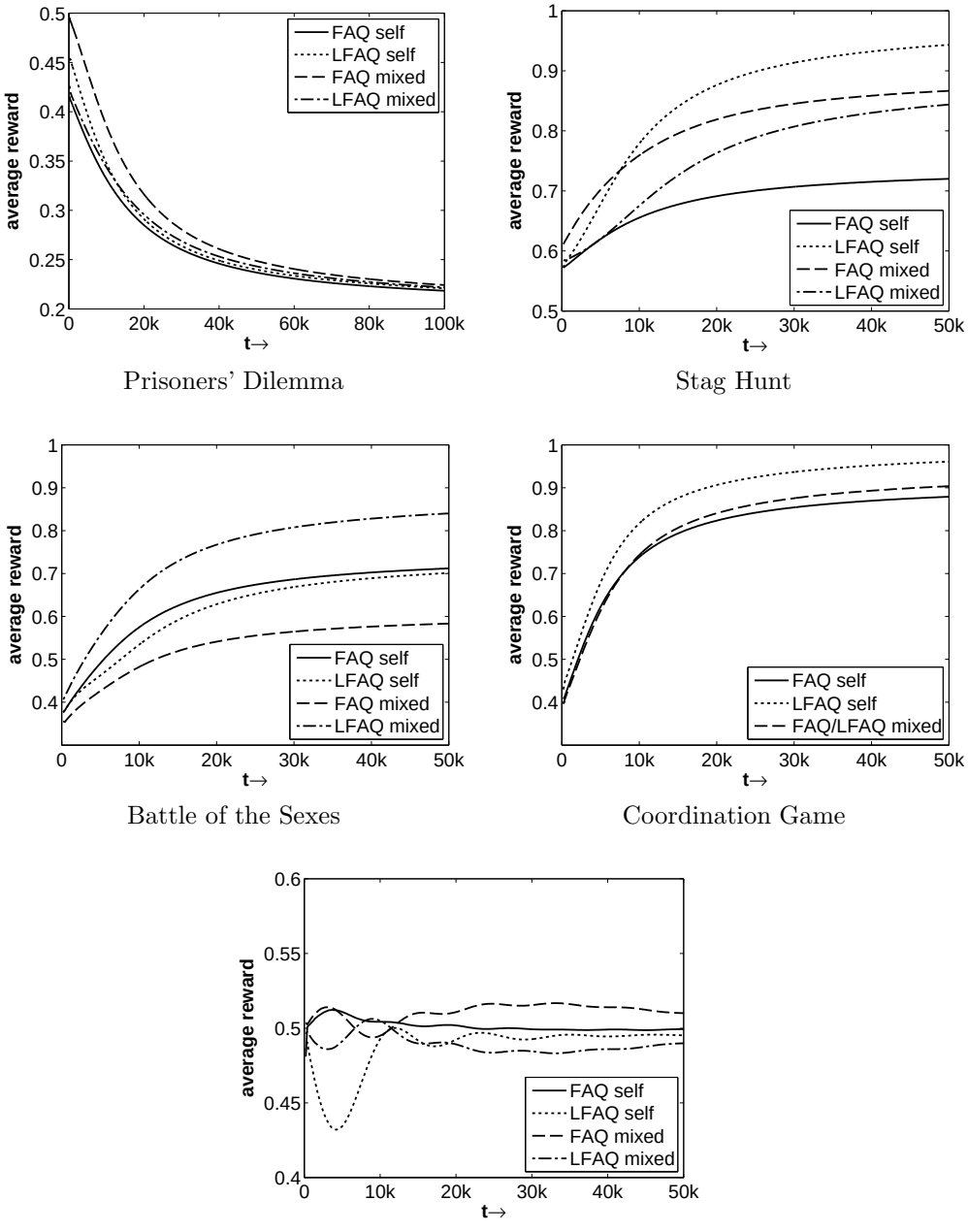
Table 6.9: Expected average reward for FAQ and LFAQ in self play and mixed play, based on the games' basins of attraction and payoff matrices.

	SH		BoS		CG	
	Player 1	Player 2	Player 1	Player 2	Player 1	Player 2
FAQ self play	0.75	0.75	0.74	0.74	0.87	0.87
FAQ - LFAQ	0.88	0.88	0.66	0.84	0.92	0.92
LFAQ self play	0.94	0.94	0.74	0.74	0.95	0.95

6.5 Summary of the results

This section provides a summary of the main results of this chapter. Note that this summary is not exhaustive, it merely serves to recapitulate several important results before proceeding with a discussion and the conclusions in the next chapter.

The proposed LFAQ algorithm is shown to match the behavior predicted by the evolutionary model derived by Panait *et al.* (2008), whereas the original Lenient Q-learning algorithm may deviates considerably. Furthermore, LFAQ behaves consistently, irrespective of the initial Q-values. These results are in line with the findings presented by Kaisers and Tuyls (2010) in the case of regular Q-learning. Furthermore, the strength of leniency in cooperative games is demonstrated, and it is argued that the proposed LFAQ algorithm inherits the theoretical claim of Panait *et al.* (2008) that a properly-set lenient learner



Prisoners' Dilemma

Stag Hunt

Battle of the Sexes

Coordination Game

Matching Pennies

Figure 6.17: Average reward over time for FAQ and LFAQ, in self play and when playing against each other.

can reach the Pareto optimal Nash equilibrium in cooperative games with an arbitrarily high probability.

Section 6.2 describes the effect of the step size α on the behavior and convergence speed of the learning algorithms. It is shown that a step size of 0.1 leads to a high variation in the results; 0.01 is found to be the maximum step size that still produces reliable results. Furthermore, a method is designed to reduce differences in the convergence speed of the algorithms in self play, by calculating a multiplication factor for the step size parameters. This is necessary in order to ensure a fair competition in the heterogeneous mixed play experiments, by ruling out artifacts based on quantitative rather than qualitative differences between the learners.

The self play experiments, described in Section 6.3, show that all learners behave as predicted by their evolutionary models. There are considerable differences between the behavior of LFAQ and the other, non-lenient, learners. This difference is most notable in common interest cooperative games such as the Stag Hunt, where LFAQ converges to the Pareto dominant equilibrium with a higher probability than the other learners, thereby achieving a higher cumulative reward.

Similar effects are observed in the mixed play experiments of Section 6.4. Mixed non-lenient learners show behavior similar to each other, and to their own behavior in self play. When LFAQ is involved, the behavior changes considerably. An interesting example is the Battle of the Sexes, where LFAQ is able to steer the learning process towards its preferred equilibrium with higher probability than the non-lenient learners. This results in a higher cumulative reward for LFAQ in mixed play, and a lower reward for its opponent. In the common interest cooperative games, LFAQ 'helps' its opponent to converge to the Pareto optimal equilibrium, which leads to a higher payoff for the opponent and a lower payoff for LFAQ itself. In general, however, LFAQ always performs at least as well against a specific opponent as any other learner does. This indicates that leniency is a dominant choice in cooperative games.

Finally, the evolutionary game theoretic approach proves effective and efficient in describing the behavior and convergence properties of reinforcement learning algorithms, both in self play and in mixed play. Moreover, based on the evolutionary models it is possible to predict the performance of the learners in a specific game, using the game's basins of attraction and payoff matrix to compute the expected average reward of the learners.

Chapter 7

Discussion and conclusions

This chapter summarizes and discusses the main findings of this thesis. After the discussion, answers to the research questions are presented, and a general conclusion to the main problem statement is given. Finally, recommendations for future work are put forward.

7.1 Discussion

This thesis contributes in four distinct ways to the field of multi-agent reinforcement learning. One, Lenient Frequency Adjusted Q-learning demonstrates how insights from evolutionary game theory can lead to new or improved reinforcement learning algorithms; two, the importance of fine tuning the learning rate is demonstrated; three, the relation between learning behavior and performance is demonstrated in homogeneous and heterogeneous environments; and four, the benefit of leniency in cooperative games is shown empirically. This section discusses these contributions in detail.

First of all, it has been shown that the improvement to Q-learning introduced by Kaisers and Tuyls (2010) applies to Lenient Q-learning as well, which led to the proposed Lenient Frequency Adjusted Q-learning algorithm. This algorithm proved to be consistent with the evolutionary model of Lenient Q-learning derived by Panait *et al.* (2008), whereas the original algorithm may deviate considerably. Furthermore, the behavior of LFAQ is independent of the initialization of the Q-values. Finally, the behavior of LFAQ is more desirable than the behavior of the original LQ algorithm with respect to the learning trajectories followed.

Second, the thorough analysis of the influence of the step size on the behavior and predictability of the learners has revealed that a step size larger than 0.01 in general leads to stochastic behavior with respect to the convergence and the trajectory of the learning process. It is important to be aware of such effects, as some applications might require more deterministic behavior. Furthermore, it has been shown how knowledge of the average convergence speed of the learners can lead to a more fair comparison, as it has been argued that such knowledge is indeed required when studying the behavior of learners in heterogeneous environments. Ignoring the underlying differences in learning speed might lead to conclusions not based on the qualitative behavior of the learners in question, but rather on their relative learning speed. A method is provided to rule out such differences, by matching the average learning speed of different learning algorithms in advance.

Third, the experiments on self play and mixed play learning provided several interesting results. First of all, the evolutionary game theoretic approach proved valuable both in self play and in mixed play. The directional field of the replicator dynamics provides a clear view on the behavior of the learners. The convergence properties of the learners in a specific game can be analyzed by calculating the game's basins of attraction, which in turn provides insight into the possible outcomes of the game. A straightforward extension of this analysis is combining the obtained basins of attraction with knowledge of the game's payoffs in order to compute the expected average reward of the learners in this game. It has been shown that these expectations agree with the simulation-based findings. An important advantage of the EGT approach is that it is computationally inexpensive. Since the replicator equations allow for exact calculations, it is not necessary to run extensive simulations to ensure statistical significance. This makes it a valuable approach for the analysis of learning behavior, which can simplify the otherwise tedious task of parameter tuning, and can facilitate the selection of a suitable RL algorithm given a certain problem.

The fourth contribution concerns the behavior of the learners, in particular the behavior of LFAQ. In self play, the lenient learner outperforms the three non-lenient learners in common interest cooperative games (the Stag Hunt and Coordination Game). In the other games, its performance is similar to that of the other learners. In mixed play, LFAQ outperforms the non-lenient learners in *all* cooperative games, also in the case of conflicting interests (Battle of the Sexes). Again, in the other games (Prisoners' Dilemma and Matching Pennies) all learners show equal performance. This confirms the findings of Panait *et al.* (2006) in homogeneous environments, and extends their work by showing that, in game theoretic terms, leniency is a dominant choice in heterogeneous cooperative games.

7.2 Conclusions

This thesis focused on the analysis of reinforcement learning algorithms by combining quantitative algorithmic performance measures with qualitative behavioral insights based on evolutionary models. Before arriving at a general conclusion, the four research questions presented in Section 1.3 are recapitulated and answered.

Research question 1. *Does the proposed Lenient Frequency Adjusted Q-learning algorithm effectively implement the evolutionary model derived by Panait* et al. *(2008), thereby resolving the discrepancies observed between the actual and predicted behavior of Lenient Q-learning?*

Lenient Frequency Adjusted Q-learning was proposed as a variation of Lenient Q-learning, based on a discrepancy between the learning trajectories and the expected dynamics of that learner. It has been argued theoretically and shown empirically that LFAQ indeed matches the evolutionary dynamics, and that its learning behavior is more preferable than that of normal Lenient Q-learning. This demonstrates that insights from evolutionary game theory can indeed be used to improve the performance of reinforcement learning algorithms.

Research question 2. *To what extent can the evolutionary game theoretic approach facilitate the analysis of multi-agent reinforcement learning algorithms in homogeneous environments, and can these insights be effectively generalized to heterogeneous environments?*

Both the simulation experiments and the evolutionary analysis indicate considerable differences in the self play behavior of lenient and non-lenient learners. These differences are most apparent in cooperative games, especially when the players share common interests. In these games, a lenient learner is able to converge to the Pareto dominant equilibrium for larger part of the policy space than a non-lenient learner, leading to a higher cumulative reward. In the two non-cooperative games, all learners show equal performance.

The results observed in mixed, heterogeneous game play show differences similar to those in self play. A combination of players that includes a lenient learner has a higher chance of converging to the Pareto optimal equilibrium in common interest cooperative games. In a cooperative game with conflicting interests, the lenient learner has a higher chance of converging to its preferred equilibrium than a non-lenient opponent. As a result, the lenient learner outperforms the non-lenient learners in all cooperative games by achieving a higher cumulative reward. In the non-cooperative games, all learners perform equally, which is expected considering that these games only have one equilibrium. Again, the evolutionary game theoretic approach enables efficient prediction of the algorithms' behavior, convergence and performance.

Research question 3. *How can the traditional algorithmic approach and the evolutionary game theoretic approach complement each other in order to analyze the link between behavior and performance of multi-agent reinforcement learning algorithms?*

It has been shown that, both in self play and in mixed play, the evolutionary game theoretic approach provides valuable insights into the expected behavior and convergence properties of (pairs of) learners. The directional field of the evolutionary dynamics can efficiently predict learning trajectories from any starting point in the policy space. Furthermore, the evolutionary dynamics allow the direct calculation of the basins of attraction of a game, given a combination of learners. This not only describes the convergence properties of these learners, but can also be used to calculate their expected average reward. These examples show that the evolutionary game theoretic approach is a valuable addition to the traditional

algorithmic approach, by providing an efficient way to predict the behavior, convergence properties and performance of multi-agent reinforcement learning algorithms.

Having answered the four research questions, it is possible to arrive at a general conclusion in relation to the initial problem statement:

Conclusion. *Lenient Frequency Adjusted Q-learning has been proposed as a new learning algorithm that implements the dynamics of the evolutionary model that was originally intended for Lenient Q-learning. Thereby, the discrepancies between the predicted and actual behavior are resolved. Furthermore, it has been demonstrated how evolutionary game theory can efficiently describe the behavior and convergence properties of reinforcement learners in homogeneous and heterogeneous environments. The expected performance of the learners can be estimated based on these results. These predictions are valuable in the successful deployment of reinforcement learning algorithms, as they provide an efficient way to select a specific learning algorithm for a given task and can guide the otherwise tedious task of parameter tuning.*

7.3 Recommendations for future work

This section highlights several possible directions for future research based on the results achieved in this thesis. First of all, it would be interesting to extend the self play and mixed play experiments to other reinforcement learning algorithms. This would make it possible to compare the theoretical findings with empirical results that can be found in literature (e.g. Bab and Brafman, 2008). At the same time, this requires the derivation of evolutionary models where they are not yet available, which in itself would be a valuable addition to the EGT approach.

Secondly, the experiments in this thesis were limited to the domain of 2×2 normal form games. Most concepts used can be translated to other domains in a straightforward manner. However, providing an intuitive visualization of games with more than two actions or players is a difficult problem. This requires a more general procedure for analyzing the behavior of a learner, that exceeds the mere visual analysis of the evolutionary dynamics.

Thirdly, an extension to multi-state environments can be considered. Although the replicator dynamics were originally intended for single-state games, recently state-coupled replicator dynamics have been proposed that show promising results in general stochastic games (Hennes, Tuyls, and Rauterberg, 2009). More elaborate analytical tools will be required in order to properly describe the behavior of these multi-state dynamics.

A fourth extension concerns LFAQ. In this thesis, a fixed degree of leniency was used for LFAQ in all experiments. It is likely that this has an influence on the results. For example, it has been shown that LFAQ outperforms non-lenient learners in cooperative games for a leniency of 5. Performing similar experiments with different degrees of leniency might provide useful insights that help in fine tuning the LFAQ algorithm.

References

Abdallah, Sherief and Lesser, Victor (2008). A Multiagent Reinforcement Learning Algorithm with Non-linear Dynamics. *Journal of Artificial Intelligence Research*, Vol. 33, pp. 521–549. [11]

Bab, Avraham and Brafman, Ronan I. (2008). Multi-Agent Reinforcement Learning in Common Interest and Fixed Sum Stochastic Games: An Experimental Study. *Journal of Machine Learning Research*, Vol. 9, pp. 2635–2675. [2, 3, 61]

Blum, Avrim and Mansour, Yishay (2007). Learning, Regret minimization, and Equilibria. *Algorithmic Game Theory* (eds. Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay Vazirani), Chapter 4, pp. 79–100. Cambridge University Press. [8]

Börgers, Tilman and Sarin, Rajiv (1997). Learning Through Reinforcement and Replicator Dynamics. *Journal of Economic Theory*, Vol. 77, pp. 1–14. [2, 4, 23, 25]

Bowling, Michael and Veloso, Manuela (2002). Multiagent learning using a variable learning rate. *Artificial Intelligence*, Vol. 136, pp. 215–250. [1, 11]

Busoniu, L., Babuška, R., and Schutter, B. De (2008). A comprehensive survey of multi-agent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, Vol. 38, No. 2, pp. 156–172. [1, 9]

Gibbons, Robert (1992). *Pearson Education.* A Primer in Game Theory. [13, 14, 15]

Gintis, Herbert (2009). *Game Theory Evolving.* University Press, Princeton NJ, 2nd edition. [2, 13, 15, 16, 17, 18, 20, 29]

Harsanyi, John C. and Selten, Reinhard (1988). *A General Theory of Equilibrium Selection in Games.* MIT Press. [16]

Hennes, Daniel, Tuyls, Karl, and Rauterberg, Matthias (2009). State-Coupled Replicator Dynamics. *Proc. of 8th Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS '09)*, pp. 789–796. [61]

Herik, Jaap van den, Kaisers, Michael, Hennes, Daniel, Tuyls, Karl, and Verbeeck, Katja (2007). Multi-Agent Learning Dynamics: A Survey. *Accepted at the Cooperative Information Agents Conference (CIA'07), Lecture Notes in Computer Science, Springer*, Delft, The Netherlands. [7, 36]

Hirsch, Morris W. and Smale, Stephen (1974). *Differential Equations, Dynamical Systems, and Linear Algebra.* Academic Press, New York. [20]

Jafari, Amir, Greenwald, Amy, Gondek, David, and Ercal, Gunes (2001). On No-Regret Learning, Fictitious Play, and Nash Equilibrium. *Proceedings of the Eighteenth International Conference on Machine Learning*, pp. 226–233, Springer. [2, 8]

Kaelbling, L.P., Littman, M.L., and Moore, A.W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, Vol. 4, pp. 237–285. [1, 5, 6]

Kaisers, Michael and Tuyls, Karl (2010). Frequency Adjusted Multi-agent Q-learning. *Proc. of 9th Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pp. 309–315. [3, 10, 25, 26, 33, 34, 39, 56, 59]

Kalyanakrishnan, Shivaram and Stone, Peter (2009). An Empirical Analysis of Value Function-Based and Policy Search Reinforcement Learning. *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, pp. 749–756. [2, 7]

Kapetanakis, Spiros and Kudenko, Daniel (2002). Reinforcement Learning of Coordination in Cooperative Multi-Agent Systems. *Eighteenth national conference on Artificial Intelligence*, pp. 326–331, American Association for Artificial Intelligence. [10]

Klos, Tomas, Ahee, Gerrit Jan Van, and Tuyls, Karl (2010). Evolutionary Dynamics of Regret Minimization. Technical report. [2, 8, 23, 25]

Maynard Smith, J. and Price, G. R. (1973). The Logic of Animal Conflict. *Nature*, Vol. 246, No. 2, pp. 15–18. [13, 16]

Nash, John F. (1950). Equilibrium Points in n-Person Games. *Proceedings of the National Academy of Sciences of the United States of America*, pp. 48–49. [15]

Panait, Liviu and Luke, Sean (2005). Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multi-Agent Systems*, Vol. 11, No. 3, pp. 387–434. [1]

Panait, Liviu, Sullivan, Keith, and Luke, Sean (2006). Lenience Towards Teammates Helps in Cooperative Multiagent Learning. *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2006)*. [10, 41, 60]

Panait, Liviu, Tuyls, Karl, and Luke, Sean (2008). Theoretical Advantages of Lenient Learners: An Evolutionary Game Theoretic Perspective. *Journal of Machine Learning Research*, Vol. 9, pp. 423–457. [5, 2, 3, 9, 10, 23, 24, 27, 34, 39, 40, 41, 56, 59, 60]

Shoham, Yoav, Powers, Rob, and Grenager, Trond (2007). If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, Vol. 171, No. 7, pp. 365–377. [1, 2]

Singh, Satinder P., Kearns, Michael J., and Mansour, Yishay (2000). Nash Convergence of Gradient Dynamics in General-Sum Games. *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI '00)*, pp. 541–548. [11]

Skyrms, Brian (2001). The Stag Hunt. *Proceedings and Addresses of the American Philosophical Association*, Vol. 75, No. 2, pp. 31–41. [15]

Sutton, R.S. and Barto, A.G. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge MA. [1, 5, 6, 7]

Thathachar, M.A.L. and Sastry, P.S. (2002). Varieties of Learning Automata: An Overview. *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, Vol. 32, No. 6, pp. 711–722. [7]

Tuyls, Karl (2004). *Learning in Multi-Agent Systems: An Evolutionary Game Theoretic Approach*. Ph.D. thesis, Vrije Universiteit Brussel. [2, 6, 8, 14, 18, 19]

Tuyls, Karl and Nowé, Ann (2005). Evolutionary Game Theory and Multi-Agent Reinforcement Learning. *The Knowledge Engineering Review*, Vol. 20, No. 1, pp. 63–90. [1, 9, 13, 15]

Tuyls, Karl, Lenaerts, Tom, Verbeeck, Katja, Maes, Sam, and Manderick, Bernard (2002). Towards a Relation Between Learning Agents and Evolutionary Dynamics. *Proceedings of BNAIC 2002, 21-22 October*, Leuven, Belgium. [2]

Tuyls, Karl, Heytens, Dries, Nowe, Ann, and Manderick, Bernard (2003a). Extended Replicator Dynamics as a Key to Reinforcement Learning in Multi-Agent Systems. *Proceedings of the 14th European Conference on Machine Learning (ECML03)*, Cavtat-Dubrovnik, Croatia. [25, 27]

Tuyls, Karl, Verbeeck, Katja, and Lenaerts, Tom (2003b). A Selection-Mutation model for Q-learning in Multi-Agent Systems. *Proceedings of AAMAS 2003, The ACM International Conference Proceedings Series*. [2, 10, 24, 26]

Tuyls, Karl, 't Hoen, Pieter Jan, and Vanschoenwinkel, Bram (2006). An Evolutionary Dynamical Analysis of Multi-Agent Learning in Iterated Games. *Journal of Autonomous Agents and Multi-Agent Systems*, Vol. 12, No. 1, pp. 115–153. [2, 23, 25]

Watkins, Christopher and Dayan, Peter (1992). Q-Learning. *Machine Learning*, Vol. 8, pp. 279–292. [1, 6, 7, 26]

Weiss, Gerhard (ed.) (1999). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambidge, MA. [1]

Wooldridge, Michael (2002). *An Introduction to MultiAgent Systems*. John Wiley & Sons, Ltd. [1, 2]

Zinkevich, Martin (2003). Online Convex Programming and Generalized Infinitesimal Gradient Ascent. *Proceedings of the International Conference on Machine Learning*, pp. 928–936. [11]