Helsinki University of Technology
Dissertations in Computer and Information Science
Espoo 2004 Report D8

MULTIAGENT REINFORCEMENT LEARNING IN MARKOV GAMES: ASYMMETRIC AND SYMMETRIC APPROACHES

Ville Könönen

Dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the Department of Computer Science and Engineering for public examination and debate in Auditorium T1 at Helsinki University of Technology (Espoo, Finland) on the 3rd of December, 2004, at 12 o'clock noon.

Helsinki University of Technology
Department of Computer Science and Engineering
Laboratory of Computer and Information Science
P.O.Box 5400
FI-02015 HUT
FINLAND

Distribution:
Helsinki University of Technology
Laboratory of Computer and Information Science
P.O.Box 5400
FI-02015 HUT
FINLAND
Tel. +358-9-451 3272
Fax +358-9-451 3277
http://www.cis.hut.fi

Available in pdf format at http://lib.hut.fi/Diss/2004/isbn9512273594/

ISBN 951-22-7358-6 (printed version)
ISBN 951-22-7359-4 (electronic version)
ISSN 1459-7020

Otamedia Oy
Espoo 2004

Könönen, V. (2004): **Multiagent reinforcement learning in Markov games: asymmetric and symmetric approaches**. Doctoral thesis, Helsinki University of Technology, Dissertations in Computer and Information Science, Report D8, Espoo, Finland.

Keywords: Markov games, reinforcement learning, Nash equilibrium, Stackelberg equilibrium, value function approximation, policy gradient

ABSTRACT

Modern computing systems are distributed, large, and heterogeneous. Computers, other information processing devices and humans are very tightly connected with each other and therefore it would be preferable to handle these entities more as agents than stand-alone systems. One of the goals of artificial intelligence is to understand interactions between entities, whether they are artificial or natural, and to suggest how to make good decisions while taking other decision makers into account. In this thesis, these interactions between intelligent and rational agents are modeled with Markov games and the emphasis is on adaptation and learning in multiagent systems.

Markov games are a general mathematical tool for modeling interactions between multiple agents. The model is very general, for example common board games are special instances of Markov games, and particularly interesting because it forms an intersection of two distinct research disciplines: machine learning and game theory. Markov games extend Markov decision processes, a well-known tool for modeling single-agent problems, to multiagent domains. On the other hand, Markov games can be seen as a dynamic extension to strategic form games, which are standard models in traditional game theory. From the computer science perspective, Markov games provide a flexible and efficient way to describe different social interactions between intelligent agents.

This thesis studies different aspects of learning in Markov games. From the machine learning perspective, the focus is on a very general learning model, i.e. reinforcement learning, in which the goal is to maximize the long-time performance of the learning agent. The thesis introduces an asymmetric learning model that is computationally efficient in multiagent systems and enables the construction of different agent hierarchies. In multiagent reinforcement learning systems based on Markov games, the space and computational requirements grow very quickly with the number of learning agents and the size of the problem instance. Therefore, it is necessary to use function approximators, such as neural networks, to model agents in many real-world applications. In this thesis, various numeric learning methods are proposed for multiagent learning problems.

The proposed methods are tested with small but non-trivial example problems from different research areas including artificial robot navigation, simplified soccer game, and automated pricing models for intelligent agents. The thesis also contains an extensive literature survey on multiagent reinforcement learning and various methods based on Markov games. Additionally, game-theoretic methods and methods originated from computer science for multiagent learning and decision making are compared.

Acknowledgments

The research work documented in this thesis has been carried out at the Laboratory of Computer and Information Science (CIS) at Helsinki University of Technology during the years 2000–2004. The research project has been funded mainly by Helsinki Graduate School in Computer Science and Engineering (HeCSE) and the CIS laboratory. I am also very grateful to Emil Aaltonen Foundation, Finnish Foundation of Technology, and Finnish Cultural Foundation for the personal grants I have received from them.

I wish to thank Academy Professor Erkki Oja for supervising my thesis and for providing excellent working facilities and conditions for me. I also wish to thank my thesis advisor, Professor Timo Honkela, for his valuable advices during the manuscript preparation.

I would like to express my sincere gratitude to the reviewers of my thesis, Dr. Kary Främling and Dr. Petri Koistinen. Their timely comments and fruitful discussions with them led to significant improvements of the manuscript. Special thanks go to Dr. Markus Koskela and Dr. Panu Somervuo for reviewing the language of the early version of the thesis.

My co-workers at CIS laboratory and my friends in general have made this period of my life very pleasant. In particular, the following persons have offered enjoyable working environment and delightful discussions: Mr. Mika Inki, Dr. Jarmo Hurri, Mr. Jarkko Salojärvi, Dr. Esa Alhoniemi, Dr. Johan Himberg, Dr. Harri Haanpää, and Dr. Sampsa Laine. Professor Esko Turunen and Dr. Jarkko Niittymäki deserve special compliments for their guidance and encouragement in the early steps of my scientific career.

I am deeply indebted to my parents, Tarmo and Hilkka, for their continuous support and encouragement during this thesis work and my whole life.

In Espoo, 15th November 2004

Ville Könönen

Ville Könönen

Contents

Abbreviations

AMG	Alternating Markov Game
CPU	Central Processing Unit
GLIE	Greedy in the Limit with Infinite Exploration
MDP	Markov Decision Process
MG	Markov Game
POMDP	Partially Observable Markov Decision Process
RL	Reinforcement Learning
RRR	Restricted Rank-based Randomized learning policies
SARSA	State-Action-Reward-State-Action an on-policy reinforcement learning method
SGP	SubGame Perfectness
TD(0)	Temporal Difference learning method with one timestep reward
VAPS	Value And Policy Search
WoLF	Win or Learn Fast
COIN	COllective INtelligence

Symbols

S	states of the environment
A^i	actions available to agent i
$r^i(s, a^1, a^2)$	reward for agent i when the actions a^1 and a^2 are selected in a state $s \in S$
s_t	state at the time instance t
a^i_t	action selected by agent i at the time instance t
r^i_{t+1}	reward for agent i associated with the state transition at the time instance t
π^i	policy function for agent i
π^i_*	equilibrium policy for agent i
$\Delta(A)$	set of all probability distributions over the elements of a set A
$([a^1], [a^2])$	strategy profile consisting of pure strategies in two-player games
$(p * [a^1] + q * [b^1],$ $r * [a^2] + s * [b^2])$	strategy profile consisting of mixed strategies in two-player games
σ^i	randomization between the pure strategies of player i (a mixed strategy)
δ	randomization between the joint strategies of the players (a correlated strategy)
$Q^i_{\pi^1,\pi^2}(s, a^1, a^2)$	total expected reward of player i when actions a^1 and a^2 are selected in a state $s \in S$ and policies π^1 and π^2 are followed thereafter
$V^i_{\pi^1,\pi^2}(s)$	total expected reward of player i in a state s when policies π^1 and π^2 are followed (value function)
γ	discount factor in reinforcement learning methods
$Q^i_t(s_t, a^1_t, a^2_t)$; $V^i_t(s_t)$	Q-function of agent i at the time instance t; value function of agent i at the time instance t
$p(s' \vert s, a^1, a^2)$	state transition probability $s \to s'$ when actions a^1 and a^2 are selected in a state $s \in S$
$P(a \vert s)$	probability of selecting action a in a state s
α_t	learning rate at the time instance t
$\mathrm{Val}\{Q^i_{\pi^1,\pi^2}(s)\}$	value of a state s for agent i when some game theoretic solution concept is applied to a Q-function $Q^i_{\pi^1,\pi^2}(s)$
$\phi(s, a^1, a^2)$	unit vector with the element corresponding to the state-actions tuple (s, a^1, a^2) set to one
$f^i(s, a^1, a^2; \boldsymbol{\omega})$	parameterized value function approximation for agent i (parameterized with $\boldsymbol{\omega}$)

$\pi(s, a^1, a^2; \boldsymbol{\theta})$	parameterized joint policy function (parametrized with $\boldsymbol{\theta}$)
$u^i(p^1, p^2)$	utility value of broker i when the current prices are p^1 and p^2 in the flat pricing model
$u^i(p^1, p^2, s; l)$	utility value of agent i when the current prices are p^1, p^2 and s in the two-layer pricing model. Maximum price limit for the brokers is l

Chapter 1

Introduction

This doctoral thesis deals with *multiagent learning* among *intelligent agents*. Here, multiagent means that there are multiple simultaneous learners present in the system; intelligence means that learners are aiming at rational decision making and the term agent refers to decentralized learning, in which the actual reasoning and learning procedures take place inside each learner.

A commonly accepted feature of intelligence is the capability of *learning*. In fact, these two terms are often linked together so strictly, perhaps too strictly, that there is a straight correspondence between intelligence and the ability to learn; a system that is capable of learning, deserves to be called intelligent and every system recognized as intelligent should also be able to learn. Overall, the goal of learning is to improve the performance of the learning agent with respect to its, perhaps unknown, environment. In this thesis, the goal of agents is to learn to anticipate long-time consequences of their action choices. This anticipation ability is achieved by using *reinforcement learning* based on *Markov Decision Processes (MDPs)*. Additionally, learning agents co-exist with each other in the environment and hence, to be able to act rationally, try to model their opponents. In this thesis, the relationship between agents is modeled with *Markov Games (MGs)* that are generalizations of MDPs for multiagent domains.

MGs can be seen as extensions to two distinct research areas: MDPs and classic (static) game theory. They extend single-agent MDPs to multiagent domains by associating a matrix game with each state of the environment. On the other hand, MGs can be seen as multistate extensions to static matrix games. Table 1.1 lists the main types of relevant optimization problems according to the number of decision makers and their time dependency. The term Markov game stems from the work of Zachrisson (1964) and they are also called stochastic games due to the earlier work of Shapley (1953). As single-agent MDPs are special cases of more general MGs, MGs are also referred to as *competitive Markov decision processes*. Single-agent MDPs are competitive Markov decision processes with zero-level of competitiveness and, in the other extreme, there are zero-sum MGs with the maximum level of competitiveness. Between these two extremes, there exist MGs where the interests of the agents are only partially conflicting.

Table 1.1: Different types of optimization problems and the corresponding research fields.

	one agent	many agents
static	decision theory	static game theory
dynamic	Markov decision processes	Markov games

The roots of MDPs and MGs extend to the 1950s. Since then, these two dynamic decision making models have evolved rather independently and have been studied in different fields such as engineering, applied mathematics, and economics. The theory of simpler single-agent MDPs evolved more rapidly and numerous efficient solving and learning methods exist for these processes. The theory of MGs has also matured significantly in recent years and some numeric solution algorithms exist for these more complex processes. However, in the case of MGs, learning the game structure and learning to play optimally are still very active research areas.

In this thesis, the focus is on efficient, both tabular and numeric, learning methods in MGs. The thesis consists of an introductory part and five publications listed in Section 1.2. The publications contain most of the contributions of the thesis and are thus referred in appropriate positions of the text.

1.1 The main contributions of the thesis

The main contributions of the thesis are:

- *Literature survey of the current status of reinforcement learning research.* The fundamental research results in the field of reinforcement learning, particularly multiagent reinforcement learning, and game theory are covered in this thesis.
- *Asymmetric multiagent reinforcement learning model.* This learning model simplifies the decision making procedure in MGs by setting an additional requirement of the ordering among decision makers. In contrast to the symmetric learning model, the asymmetric learning model has stronger convergence properties and lower computational and space requirements.
- *Numeric methods for value-function-based and policy-gradient-based multiagent reinforcement learning in MGs.* Applying multiagent reinforcement learning methods to realistic problems requires the use of function approximators such as neural networks. For this purpose, efficient and general gradient based methods for multiagent reinforcement learning are proposed in this thesis.
- *Hybrid model for multiagent reinforcement learning in MGs.* In MGs, the opponents are modeled in each state of the system by using matrix games. In many problem instances this is not required and hence, in this thesis, a method is proposed for dividing the state space into two subspaces: the space of complex states and the space of simple states. The opponent is modeled only in the complex states resulting in much lower computational and space requirements than in standard MGs.

1.2 Publications and author's contributions

The thesis includes the publications listed below. The corresponding Roman numbering of the publications is used through the thesis when referring to the publications.

I Könönen, V. J. (2004). Asymmetric multiagent reinforcement learning, *Web Intelligence and Agent Systems: An International Journal (WIAS)* **2**(2): 105–121.

II Könönen, V. J. (2004). Gradient Descent for symmetric and asymmetric multiagent reinforcement learning, *Technical Report A78*, Helsinki University of Technology, Publications in Computer and Information Science.

III Könönen, V. J. (2004). Hybrid model for multiagent reinforcement learning, *Proceedings of the International Joint Conference on Neural Networks (IJCNN-2004)*, Budapest, Hungary, pp. 1793–1798.

IV Könönen, V. J. (2004). Policy gradient method for team Markov games, *Proceedings of the Fifth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL-2004)*, Exeter, UK. pp. 733–739.

V Könönen, V. J. and Oja, E. (2004). Asymmetric multiagent reinforcement learning in pricing applications, *Proceedings of the International Joint Conference on Neural Networks (IJCNN-2004)*, Budapest, Hungary, pp. 1097-1102.

In Publication I, the author of the thesis proposes a multiagent reinforcement learning model based on the asymmetric (Stackelberg) equilibrium concept. The publication also presents an off-policy learning method that utilizes the proposed learning model. Moreover, the convergence properties of the proposed method are studied analytically and empirically.

In Publication II, the author extends the general numeric gradient-based learning framework from single-agent domains to multiagent domains. The framework presented in this publication takes into account the possible time varying exploration policy and makes it is easy to apply different error criteria. The proposed method is tested with two example problems.

In Publication III, the author proposes a method that combines normal single-agent and multiagent reinforcement learning methods by dividing the state space of the problem into two subspaces: one containing states where the other agents are modeled and the other where the opponents are not modeled. The state space division is done by utilizing task specific information. The proposed method significantly reduces the computational and space requirements of the multiagent reinforcement learning methods based on MGs. Moreover, the proposed method is tested with a small example problem.

In Publication IV, the author applies asymmetric multiagent reinforcement learning to team MGs. Based on this learning method, the direct policy gradient method, originally developed for single-agent learning tasks, is extended to team problems. In addition, the proposed method is applied to a simple soccer game.

In Publication V, asymmetric multiagent reinforcement learning is applied to two pricing problems. The original idea of using asymmetric learning to pricing problems was developed by the author. The systems used in testing the learning models were implemented by the author. Both authors participated in writing the article.

1.3 Organization of the thesis

The rest of this thesis is organized as depicted in Fig. 1.1. Chapter 2 contains background and basic principles of reinforcement learning based on MDPs. Chapter 3 concentrates on basic concepts and foundations of game theory. These two chapters are mostly parallel and aimed to form two distinct foundations for the next chapter on multiagent reinforcement learning in MGs. Chapter 4 is also the main chapter of the thesis containing, together with Chapter 5 and publications, most of the contributions of the author. Chapter 5 introduces three example applications of multiagent reinforcement learning based on MGs. Finally, Chapter 6 concludes the thesis.

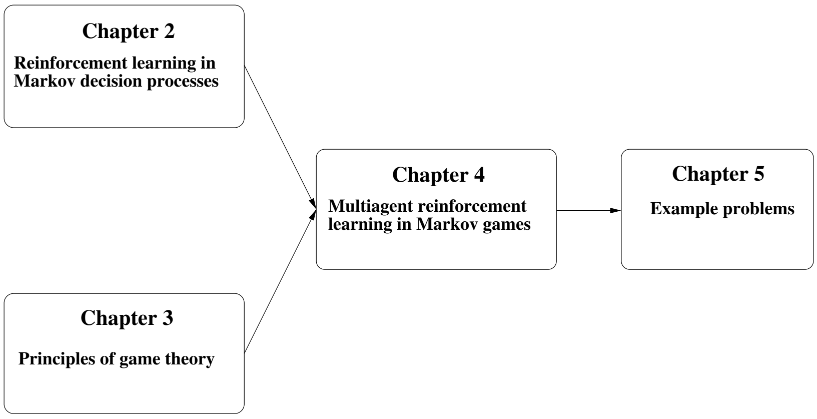

Chapter 2
Reinforcement learning in Markov decision processes
Chapter 3
Principles of game theory
Chapter 4
Multiagent reinforcement learning in Markov games
Chapter 5
Example problems

Figure 1.1: Mutual relationship of the chapters in the thesis.

Overall, this work has been written from the computer science point of view. The used notation may differ, in place to place, from the usual notation in game theory. In particular, the decision makers are commonly referred to as *players* and their options as *strategies* in game theory whereas in machine learning as *agents* and *actions*, respectively. In this thesis, the decision makers face a game theoretic problem at each time step, usually identified as a stagegame, and therefore game theoretic terms are used when discussing these stagegames. On the other hand, when learning processes containing multiple stages are discussed, machine learning terms are utilized.

Chapter 2

Reinforcement learning in Markov decision processes

There are basically three major learning paradigms in machine learning: *supervised learning*, *unsupervised learning* and *reinforcement learning (RL)*. In supervised learning, there exists a teacher having knowledge of the environment, in the form of input-output pairs, and the learning system for which the environment is unknown. The teacher provides samples from the environment by giving correct outputs to inputs and the goal of the learning system is to learn to emulate the teacher and to generalize the samples to unseen data. In unsupervised learning, contrary to supervised learning, there exists no external teacher and therefore no correct outputs are provided. RL is located between supervised and unsupervised learning: correct answers are not directly provided to the learning system but it learns features of the environment by continuously interacting with it. The learning system takes actions in the environment and receives reward signals from the environment corresponding to these action selections.

In this thesis, the goal is to study RL in systems with multiple simultaneous learners in the same environment. The methods discussed in the thesis lean heavily on single-agent RL methods. This chapter provides a short view to the history of single-agent RL and to the general principles of single-agent RL methods. Moreover, the focus is on problems in which the consequences (rewards) of selecting an action can take place arbitrarily far in the future. The mathematical tool for modeling delayed reward problems are Markov Decision Processes (MDPs) and thus the methods discussed here are based on MDPs.

2.1 History of RL in brief

The modern RL methods have their roots in the early 1910s. American psychologist Edward Thorndike made studies on animal psychology and the psychology of learning. In Thorndike (1911), he discussed the idea that the actions producing satisfaction to the animal will be later reselected more likely than the actions producing discomfort. This simple principle is called the *Law of Effect* and it is, due to its intuitivity, widely regarded as a basic principle of human and animal behavior (see e.g. Bower and Hilgard, 1981; Cziko, 1995; Dennett, 1978).

Another major research discipline that has greatly affected modern RL is optimal control theory. It emerged in the late 1950s mainly due to the work of Bellman (see e.g. Bellman, 1957a; Bellman, 1961). The goal of optimal control theory is to find a control that minimizes some measure of the system's behavior over time. A general tool for solving optimal control problems is dynamic programming. Bellman also proposed an MDP, the core of most RL methods, as a model of discrete and stochastic optimal control problem in Bellman (1957b).

These two branches joined in the late 1970s to produce an efficient learning technique, namely *temporal difference learning*. Its roots are in the concept of *secondary reinforcers*, i.e. reinforcements that are paired with *primary reinforcers*, like food or pain. For example, the well-known work of Pavlov (1984) (unabridged edition of the original text published in 1927) provides an example of secondary reinforcers in which a sound is the secondary reinforcer and the food is the primary reinforcer. In temporal difference learning, the learning agent learns to anticipate long-time consequences of its action choices. In this case, long-time expected rewards can be considered as secondary reinforcers and the immediate rewards as primary reinforcers. The first paper proposing temporal difference learning rules was by Witten (1977). Thereafter, numerous articles on temporal difference methods have been published, e.g. the *actor-critic architecture* by Sutton, Barto and Anderson (1983). Perhaps the most well-known methods for solving reinforcement learning problems are *Q-learning* (Watkins, 1989) and *SARSA-learning* (Rummery and Niranjan, 1994). The convergence of these methods were studied in Watkins and Dayan (1992) and Singh, Jaakkola, Littman and Szepesvári (2000). A thorough survey of the basic methods for RL problem can be found in Kaelbling, Littman and Moore (1996).

2.2 General principles

A fundamental concept in RL is an *agent* that interacts with its environment in the manner illustrated in Fig. 2.1. The agent is assumed to be *autonomous* and *rational*. Autonomous means that the agent is capable of making decisions on its own and everything else, i.e. its real environment, other agents, etc., are part of its environment. The agent also has a purpose or a goal, perhaps set by the designer of the agent and so it has a utility function representing its goal. It tries to behave rationally with respect to this utility function, i.e. to select actions that maximize its expected utility value.

Because the agent is an autonomous entity, it should have some instruments for sensing and affecting its surrounding environment. The agent has *sensors* for observing the environment and *effectors* for changing the environment by its own action selections.

2.2.1 A formal model of the agent based system

At each time step, the environment is in some state. The agent knows this state, either exactly or partially, and makes its action choices based on this information. The agent then

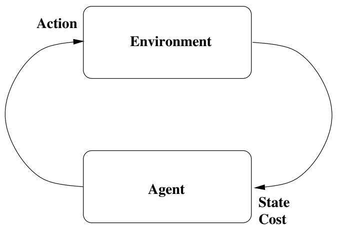


Figure 2.1: An overview of the learning system in its environment.

implements its action selection by using the effectors and based on the action selection, the environment changes its state, perhaps stochastically. After the state transition, the agent observes the new state and an immediate reward associated with the state transition.

Formally, the model of the system consists of the following parts (in this thesis it is assumed, for brevity, that the system is discrete, i.e. time, states, and actions are all discrete):

- The environment is in some discrete state $s \in S$ at each time step.
- The learning agent has a set of actions available in each state $s \in S$, denoted $A(s)$.
- When the system changes its state, the agent gets a real-valued reward $r \in \mathbb{R}$.
- There is a function π that stipulates the behavior of the agent and thus summarizes the agent's knowledge of its environment.

The main difference between supervised learning and reinforcement learning is that in reinforcement learning, the feedback from the environment does not include information about the right action choice. It only provides a punishment or a reward signal based on the current action choice of the agent. Due to the lack of correct answers, the learning system should gather information of its environment in a trial-and-error manner. This leads to the exploration vs. exploitation dilemma that will be discussed at the end of this chapter.

2.2.2 Suitable performance criteria

A rational agent always selects the action that maximizes some performance criterion that reflects design objectives of the agent. Usually it is not enough to maximize only the direct reward r. Instead, it is preferable to maximize the long time performance of the agent. Three suitable long-time performance criteria are presented below. In each case, the criterion is an expected value because different sources of stochasticity can exist in the system (state transitions, action choices, etc.). Moreover, all criteria depend on the policy π that stipulates the behavior of the agent.

- Expected total reward: $E_\pi[\sum_{t=0}^{h} r_{t+1}]$. In this case, the objective function is the total, cumulative reward collected in some finite length h episode of the interaction between the agent and the environment. This error criterion is suitable for the cases in which the length of the task is known to the agent a priori. However, the difficulty is that the optimal behavior (policy) is not necessarily stationary; it could change over time.

– Expected total discounted reward: $E_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t+1}]$. This is almost the same as the previous criterion except that now the horizon h is infinite. With infinite horizon, the expected total reward would be unbounded and therefore the rewards are discounted by a discount factor $\gamma \in [0, 1[$ that controls the balance between the significance of immediate rewards and future rewards. If γ is near zero, the agent makes its decisions almost myopically; decisions are based on immediate rewards and, on the other hand, if γ is near one, the agent is willing to sacrifice immediate rewards for acquiring a large long-time expected utility value.

– Expected average reward: $\lim_{h\to\infty} E_\pi[\frac{1}{h}\sum_{t=0}^{h} r_{t+1}]$. In this case, the goal is to maximize the total average reward over time. This criterion is complementary to the previous performance criterion and, in fact, the optimal policy maximizing the expected discounted reward criterion becomes equivalent to the policy maximizing the expected average reward criterion when γ approaches unity. The main difficulty with this criterion is that it is not possible to balance between policies that emphasize short time rewards and then policies that emphasize longtime rewards. On the other hand, the methods that use the expected average reward criterion do not have a discount factor parameter γ and thus have one parameter less than methods based on the expected discounted reward criterion.

The second performance criterion, the expected total discounted reward criterion, is the one used most in reinforcement learning literature. This is mainly due to the fact that it is easier to handle mathematically than the other two criteria. All methods presented in this thesis utilize this performance criterion.

2.3 Markov decision processes

The actual consequences of an action selection in the current state may take place in the distant future. Therefore it is very important to be able to anticipate these consequences and always choose an action that leads to optimal behavior. This problem of delayed rewards is often called the *credit assignment problem* and the tool for handling it is an MDP. A thorough coverage of the basic principles of MDPs can be found in (Sutton and Barto, 1998).

Formally, a Markov decision process can be defined in the following way:

Definition 2.1 *A* Markov Decision Process *(MDP) is a tuple (S, A, p, r), where S is the set of all states, A is the set of all actions, $p : S \times A \to \Delta(S)$ is the state transition function and $r : S \times A \to \mathbb{R}$ is the reward function. $\Delta(S)$ is the set of all probability distributions over the set S.*

At each time step, the environment is in some state $s \in S$ and the agent selects an action $a \in A(s)$. Based on this action selection, the environment changes its state according to the probability distribution p and the agent gets an immediate reward r. The state transition probabilities p may depend only on the current state of the environment and the current action selection of the agent. In this thesis, it is assumed for brevity that the agent has the same set of actions available in each state, i.e. $A(s) = A, \forall s \in S$.

The central concept in MDPs is the policy function π that stipulates the actual behavior of the agent, i.e. it maps states to actions. Formally, the following mapping exists for all t:

$$\pi : S_t \rightarrow A_t. \tag{2.1}$$

Note that this policy function depends also on time, i.e. the function can return different actions at different time steps t even in the same state s. This property is needed in the finite horizon problems where the optimal policy can be time dependent. For example, consider a game playing situation where the player (agent) knows the actual length of the game. In the last rounds of the game, the player might be willing to take more risks to achieve his goals than in the beginning of the game if he seems to be losing the game. In infinite horizon problems or problems in which the actual length of the task is not known a priori, the only way to behave optimally is to assume at each time step that the task will still continue infinitely long and therefore the optimal policy is not time-dependent, i.e. the policy π is *stationary*:

$$\pi : S \rightarrow A. \tag{2.2}$$

In this thesis, the focus is on stationary policies.

For making decisions in MDPs, the agent needs to associate values for the states of the environment. A natural choice for this value is one of the above presented performance criteria when the policy π is followed. In the case of the expected total discounted utility value, this criterion takes the following form:

$$V_\pi(s) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s \right], \tag{2.3}$$

which is simply the expected (state transitions are generally stochastic) discounted sum of rewards when the agent starts from the state $s \in S$ and follows the policy π infinitely. For the actual comparison of the actions in an arbitrary state $s \in S$, it is useful to extract the immediate rewards from $V_\pi(s)$. This function is often called Q-function and formally it can be expressed in the following way:

$$Q_\pi(s, a) = E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, a_0 = a \right]. \tag{2.4}$$

This can be written in the following recursive form that is also known as the *Bellman equation*:

$$Q_\pi(s, a) = r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V_\pi(s'). \tag{2.5}$$

The literal interpretation of the above equation is that the value of selecting an action a in a state s is the sum of the immediate reward r and the discounted expected value of further states when the policy π is followed.

Bellman's optimality principle states that no matter how the agent has reached the current state it should behave optimally from now on. Applying this principle simultaneously in each state $s \in S$, the global optimal behavior is acquired. Hence, for the optimal policy π_* it should hold in every state $s \in S$ that

$$V_{\pi_*}(s) = \max_{a \in A} \left[r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V_{\pi_*}(s') \right]. \tag{2.6}$$

The policy function is a mapping from the state space S into the action space A. For the rational agent and for the optimal value function V_{π_*}, the optimal policy function can be written in the following form:

$$\pi_*(s) = \arg\max_{a \in A} Q_{\pi_*}(s,a). \tag{2.7}$$

Note that because the optimal policy function implements the rational agent that maximizes its expected discounted utility in each state, the value of each state is always the value of the maximizing action. Therefore, it is possible to write Eq. (2.6) in the policy independent form:

$$V_*(s) = \max_{a \in A} \left[r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V_*(s') \right]. \tag{2.8}$$

In this thesis, the subscripts denoting optimal policies are preferred because in the following chapters, where multiagent extensions to MDPs are discussed, it is easier to distinguish between different agents and their policies by using these subscripts.

2.4 Actual solving and learning methods for MDPs

In this section, several basic solution and learning methods for MDPs are discussed briefly. If the model of the environment, i.e. state transition probabilities and rewards r are known, all information about the MDP is known a priori and it is possible to solve the problem by using various available solution methods. If this model is not known, there are two possibilities (Kaelbling et al., 1996):

1. Learn a model of the environment by interacting with the environment and then solve the problem by using some solution method. These methods are usually referred to as *model-based* methods.
2. Learn the optimal value function by interacting with the environment and then act optimally with regard to this value function. These methods are often called *model-free* methods.

An overview of different methods is provided in this section. However, the focus of this thesis is on model-free learning methods and therefore only the relevant solving and learning methods are discussed in detail.

2.4.1 Solving methods for MDPs

It is possible to find the unique optimal value function V_{π_*} by solving simultaneously Eq. (2.6) for each state $s \in S$. Unfortunately the equation contains the maximum operator and hence it is not possible to solve the problem by using standard methods for systems of linear equations such as Gaussian elimination. In addition, if the state space of the problem is large, solving Eq. (2.6) for each state becomes intractable. However, it is possible to express the problem as a linear program and solve it by using one of the many methods designed for solving linear programs, e.g. the *Simplex method* due to Dantzig (1963). A lot of software packages and subroutine libraries also exist for this purpose (see e.g. NAG, 2004; IMS, 2004). However, also in this case, the number of linear constraints is quite large even with relatively simple problems.

The *policy iteration* and *value iteration* algorithms are two very simple and important algorithms for solving MDPs. Also, many more sophisticated learning methods are based on the ideas behind these two algorithms. The main idea behind the policy iteration algorithm is to improve the policy function π directly. The algorithm calculates the value function corresponding to this policy function by solving the system of linear equations and then determines the greedy policy according to this value function. The algorithm stops when there are no changes in the policy function in two consecutive iterations of the algorithm. If $|\cdot|$ denotes the size of an arbitrary set, there are $|A|^{|S|}$ distinct policies and since the policy improves in each iteration, the algorithm requires at most $|A|^{|S|}$ iterations (Littman, 1996). The policy iteration algorithm is presented in the algorithmic form in Algorithm 2.1.

Solving the system of linear equations in the policy iteration algorithm requires a lot of computation if the state and the action spaces are large. Another way to solve MDPs is to use the value iteration algorithm which is based on successive approximations of the value function V and thus there is no need to compute the exact value of this value function in each state in every iteration of the algorithm. The value iteration algorithm is listed in Algorithm 2.2.

In Algorithm 2.2, the end condition that is expressed in a very rough manner as the *policy is good enough*, must be selected suitable for the problem in hand. A natural choice is to calculate distances between vectors containing values of V in each state from consecutive iterations of the algorithm and terminate the algorithm when the difference is small enough. However, in many problems, particularly in control problems, only mutual ordering of the actions counts and this ordering converges much before the value function. Therefore some other stopping criteria may be used.

2.4.2 Learning methods for MDPs

If the model of the environment is not known a priori, it must be estimated during the learning process. In the value iteration algorithm, the whole state space is swept through in each iteration and the real model of the environment is used to estimate the Q-function. In the *TD(0)-algorithm* by Sutton (1988), the value function is updated by using data gathered from the interaction with the environment. When the environment is in a state s_t at time instance t and the agent selects the action a_t, the system changes its state to s_{t+1} and the

Algorithm 2.1 Policy iteration algorithm.

```
Initialize the value function $V$ arbitrarily, e.g. $V(s) = 0$ for each $s \in S$. Initialize the
policy function $\pi$ arbitrarily, i.e. $\pi(s) \in A(s), \forall s \in S$.
*policy evaluation*:
**repeat**
  **for all** $s \in S$ **do**
    $V_{t+1}(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s'|s, \pi(s)) V_t(s')$
  **end for**
**until** maximum change in $V$ is $\leq \epsilon$
*policy improvement*:
$\pi_{\text{old}} = \pi$
**for all** $s \in S$ **do**
  $\pi(s) = \arg\max_{a \in A} [r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V(s')]$
**end for**
**if** $\pi_{\text{old}} <> \pi$ **then**
  Go to *policy evaluation*
**end if**
```

Algorithm 2.2 Value iteration algorithm.

```
Initialize the value function $V$ arbitrarily, e.g. $V(s) = 0$ for each $s \in S$.
**while** policy is not good enough **do**
  **for all** $s \in S$ **do**
    **for all** $a \in A$ **do**
      $Q(s, a) = r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) V_t(s')$
    **end for**
    $V_{t+1}(s) = \max_{a \in A} Q(s, a)$
  **end for**
**end while**
```

agent gets the reward r_{t+1}. Then, the value function V is updated by using the following equation:

$$V_{t+1}(s_t) = (1 - \alpha_t) V_t(s_t) + \alpha_t [r_{t+1} + \gamma V_t(s_{t+1})]. \tag{2.9}$$

The main idea of this algorithm is to move the value of $V(s_t)$ gradually toward the expected value of $r_{t+1} + \gamma V(s_{t+1})$ which is the desired value of $V(s_t)$. Because the agent observes only one state transition at each time step, the algorithm is a stochastic approximation algorithm and therefore the learning rate parameter should satisfy the usual conditions of stochastic approximation theory, see Robbins and Monro (1951). Note that the TD(0) algorithm is, in fact, a version of the *Widrow-Hoff* learning rule (Widrow and Hoff, 1960) in which the desired output is set to $r_{t+1} + \gamma V(s_{t+1})$.

Perhaps the most used algorithm for reinforcement learning is Q-learning, which is a version of the above learning rule in which the value function V is substituted with the Q-function and the policy is the greedy policy with respect to the Q-function. The actual learning rule takes the following form:

$$Q_{t+1}(s_t, a_t) = (1 - \alpha_t) Q_t(s_t, a_t) + \alpha_t [r_{t+1} + \gamma \max_{b \in A} Q_t(s_{t+1}, b)]. \tag{2.10}$$

Particularly in control problems, a fitness value should be available for each state-action pair, which is often difficult to implement with TD(0)-learning rule. In Q-learning, the agent can select its actions directly based on Q-function in each state and when the process has almost converged it is reasonable to act rationally and select the actions that maximize the Q-function in each state. Another advantage of Q-learning is that it is *exploration insensitive* in the sense that it is guaranteed to converge if every state-action pair is visited infinitely often. For this reason, Q-learning is often referred to as an off-policy method. As state and action spaces of real-world problems are often very large, this property is seldom satisfied and the actions are often selected by using some of the exploration techniques discussed at the end of this chapter.

SARSA-learning differs from Q-learning in the sense that it does not use an artificial action in the state s_{t+1}, i.e. an action that maximizes the Q-function. Instead, it selects actions based on the current, possibly non-stationary, exploration policy. The update rule for the SARSA-learning is as follows:

$$Q_{t+1}(s_t, a_t) = (1 - \alpha_t)Q_t(s_t, a_t) + \alpha_t[r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1})]. \tag{2.11}$$

This learning rule can learn the optimal Q-function if the used exploration policy chooses actions optimally in the limit. This property is satisfied in the *GLIE (Greedy in the Limit with Infinite Exploration)* exploration policy. Although the use of on-policy learning methods such as SARSA, set additional restrictions to the exploration policy, on-policy methods appear to be superior to off-policy algorithms in many prediction and control problems (Singh, Jaakkola, Littman and Szepesvári, 2000).

TD(0), Q-learning and SARSA-learning build the model of the environment implicitly during the learning process. Another approach is to build the model during the learning process and then use this model for value or Q-function updating. Examples of this type of learning methods are DYNA proposed in Sutton (1990), Sutton (1991) and prioritized sweeping proposed by Moore and Atkeson (1993). Additionally, the specific category of model based methods contain methods that utilize normal on-policy or off-policy learning methods but also predict the parts of the environment that are useful for the agent (see e.g. Barto, Bradtke and Singh, 1995; Dean, Kaelbling, Kirman and Nicholson, 1993). The basic solution methods for RL problems are discussed by Sutton and Barto (1998). More mathematical coverage of RL methods with an emphasis on the stochastic-approximation theory is given by Bertsekas and Tsitsiklis (1996). A broad perspective to machine learning is given by Mitchell (1997).

2.5 Exploration vs. exploitation

In real-world applications of RL, state and action spaces become very large and therefore it is intractable to visit all states or state-action pairs multiple times during the learning process. The one way to circumvent this problem is to interact with the environment and select the actions that are expected to lead to the most interesting and most useful parts of the state space. The decisions are often based on the current estimation of the value function and some probabilistic component that occasionally selects actions that do not have high expected utility values yet. If this random component is too small, some good areas of the state space may remain unvisited and the learned value function is suboptimal. On the other hand, selecting all actions at random is too inefficient.

Perhaps the most used exploration method is ϵ-*greedy* exploration in which actions are usually selected by using greedy action selection but with a small probability, ϵ, actions are selected from the uniform distribution independently from the Q-function. In the beginning of the learning process ϵ is usually very large and it is decayed during the learning.

Another well-known method is the *softmax* action selection in which the action selection probabilities are determined by the Gibbs distribution in the following way:

$$P(a|s) = \frac{e^{\beta Q(s,a)}}{\sum_{b \in A} e^{\beta Q(s,b)}}, \tag{2.12}$$

where β is the temperature parameter the determines how random the action selection procedure is. If β is near to zero, the agent selects actions almost randomly and when β approaches infinity, the action selection approaches greedy action selection. This method together with ϵ-greedy action selection results in GLIE exploration methods because they approach greedy action selection eventually.

Another class of exploration methods is *RRR (Restricted Rank-based Randomized learning policies)* exploration. In RRR exploration methods, the actions are ranked according to some rule and then the ranks determine actual probabilities for choosing actions so that higher ranks get the same or higher probability than lower ranks. Different exploration strategies can be represented in the RRR framework, e.g. pure random exploration or the ϵ-greedy exploration. Note that softmax action selection depends directly on Q-values and therefore it is not an RRR exploration policy.

Chapter 3

Principles of game theory

Mathematical game theory provides means to formally inspect social interactions between *rational* and *intelligent* decision makers. In game theory, the term *game* refers not merely to literal games even though it is possible to inspect also these games with game theory, but any social interactions between two or more decision makers. Social interactions consist of any kind of cooperation or conflict situations between humans or any other type of autonomous agents. Game theoretic models are used to analyze these interactions in various domains including finance, politics, and warfare.

Decision makers in game theory are usually denoted as *players* and the options available to the players as *strategies*. The basic assumption is that the players are intelligent and rational. Intelligence means that all players know everything that the game theorist knows and are thus capable of making the same inferences as the game theorist. Rationality means that players always try to maximize their social welfare. As the modern game theory is built on decision theory, the social welfare of the players is evaluated in terms of utility value and therefore rational players select strategies that maximize their expected utility values taking into account the strategies of their opponents.

The main aim of this chapter is to provide a brief outline of game theory, its history and a review of methods needed in the consecutive chapters. The discussion is divided into four sections. In the first section, the history of game theory is discussed in brief. Mathematical games can be expressed in different forms. An overview of these forms and the relationships between them is given in the second section. When the social interaction between players is expressed as a mathematical game, the goal of the players is to find a solution to the game, i.e. strategies that maximize their expected utility values. Suitable solution concepts are discussed in the third section. Most of these solution concepts are simply special cases of the *Nash equilibrium concept*, which is perhaps the most studied solution concept in game theory. If some special properties are required from the applied solution concept, the game can be represented in a more compact and computationally efficient form. These issues are covered in the last section of the chapter. In all examples presented in the chapter, numbers denote payoff values for the players and the goal of the players is to maximize their payoffs.

3.1 Brief history of game theory

Modern game theory can be said to have been established by Morgenstern and von Neumann in their seminal book (von Neumann and Morgenstern, 1944). However, earlier work by Cournot (1897), Edgeworth (1881), Zermelo (1913), Borel (1953c), Borel (1953a), Borel (1953b) (English translations from original articles published in the 1920s), and von Neumann (1959) can be thought to construct a prelude for this seminal book and thus the whole theory of mathematical games. Cournot studied economical systems and discussed a special case of duopoly and a solution concept that is, in fact, a special case of the Nash equilibrium concept. Edgeworth studied the role of agreements in free markets and proposed the concept of contract curve as a solution to trading between individuals. Zermelo studied the game of chess and provided a proof that the game has an optimal strategy, corresponding to a subgame perfect equilibrium that is a Nash equilibrium solution of chess. Borel proposed the concept of mixed strategy, i.e. a randomization between two or more pure strategies in a game. In addition, he proposed the minimax solution concept in two-person zero-sum games and proved its existence in a few special cases. Von Neumann then extended this existence proof for arbitrary two-player zero-sum games. Another solution concept for general-sum mathematical games was proposed by von Stackelberg (1934) within the context of economic competition. English translation of this work can be found in von Stackelberg (1952).

In von Neumann and Morgenstern (1944), the authors considered only *minimax solutions* proposed earlier by Borel and von Neumann in von Neumann (1959) and Borel (1953c), Borel (1953a), Borel (1953b), respectively. In 1950, Nash extended this solution concept for general-sum games in Nash (1950b). The Nash solution concept, providing a safe solution to general-sum games from which no single player is willing to deviate, is perhaps the most studied solution concept in game theory. In fact, most of the solution concepts proposed later are only special cases of Nash equilibria. Nash also studied bargaining problems in Nash (1950a) and Nash (1953). He was awarded the Nobel prize in economy for his contributions in game theory in 1994 together with Harsanyi and Selten.

Kuhn (1953) introduced extensive form games with imperfect information (extensive form games were introduced earlier by von Neumann and Morgenstern (1944)). Selten (1975) proposed a suitable solution concept for extensive form games, the *subgame perfect equilibrium concept*. A wide range of models of repeated games, i.e. static games that are repeated finitely or infinitely many times, have been studied in the game theory literature. Perhaps the most interesting special case is the *stochastic game model (Markov game model)* originally proposed by Shapley (1953) that is, in fact, a natural game theoretical extension to single-agent Markov decision processes originally proposed by Bellman (1957a). Markov decision processes and Markov games have a close relationship, as was discussed in the previous chapter, to optimal control theory and its multiplayer version, dynamic game theory.

3.2 Basic game theoretical models

Mathematical games can be represented by using two different main representations: the *strategic form* and the *extensive form*. In this section, mathematical games are categorized with respect to two attributes: the number of players in the game and the correlation between the payoff values of the players. In addition, the two main representations of the games are discussed and their features are compared. The thorough discussion of the basic representations and their mutual relationships can be found in Myerson (1991).

3.2.1 Categorization of games

Modern game theory can be considered as an extension of decision theory to the situation which contains multiple decision makers. Thus, if the game only has one decision maker, it can be handled as a decision theoretic problem as the player always implements a strategy that maximizes the player's own expected utility function. As the number of players increases, the computation of a solution becomes, depending on the solution concept, more demanding.

Another important property of games is the payoff correlation between players. With a maximal correlation, either positive or negative, in the payoff values of players, the solution of the game is easier to obtain than in the general case; it can be said that the operators are more "close" to the normal maximum operator of decision theory. In Fig. 3.1, an illustration of how the nature of a mathematical game depends on its payoff structure is presented. In a team game, the same utility function is shared by all players and hence the correlation between players' payoff values is at maximum. On the other end, in zero-sum games the payoffs are complements of each other and there exists the maximal negative correlation between the payoff values. In general-sum games, the objectives of the players are partially conflicting and therefore these games lie between the extreme cases presented in Fig. 3.1.

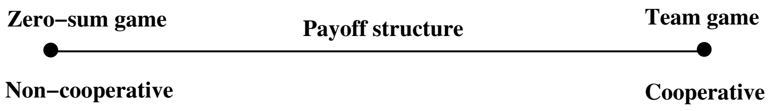
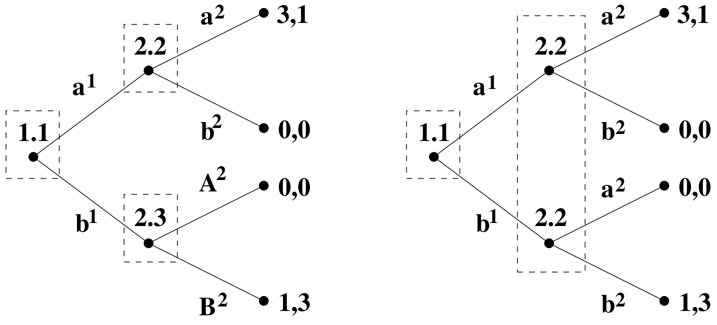


Figure 3.1: Payoff structure vs. nature of games.

3.2.2 Extensive form

Extensive form is the most versatile way to represent social interactions between decision makers. It models the temporal structure of the game explicitly and therefore even large board games can be represented as extensive form games.

In Game 3.1, there are two simple examples of extensive form games. An extensive form game is a tree that represents all the events that can occur in the game. The tree consists of *nodes*, one of which is the *root* that represents the beginning of the game. Each node is controlled by one of the players, i.e. one of the players makes its strategy selection in each node. The nodes are connected with *branches* that represent different strategy choices for the players. The nodes that are not followed by any further branches are *terminal nodes* and they represent possible ways that the game could end. At each terminal node, there is a set of numbers, separated by commas, that represent payoff values for each player when the game ends at this terminal node. When the game is actually played, the game instance forms a path from the root node to one of the terminal nodes; this sequence of events is called the *path of play.* Each node has two labels: the first corresponds to the number of the player that controls the node and the second to the information set of the node. If two nodes are in the same information set, the player that controls these nodes can not distinguish between these decision nodes. In Game 3.1, the two games differ only in that player 2 has different information on the strategy selected by player 1. In the game on the left, both nodes controlled by player 2 have distinct information states and therefore player 2 knows what strategy player 1 has selected at the previous time step. In the game on the right, player 2 can not distinguish between the two nodes and therefore has no information about the strategy selection of player 1. In fact, the rightmost game illustrates the situation where both players choose their strategies simultaneously. In addition, extensive form games may have probabilistic change nodes that are usually marked with the player number 0 ("Nature") and strategies available for this player marked with probability values.



Game 3.1: Two extensive form games with different information states for player 2.

3.2.3 Strategic form

A simpler and more compact way than the extensive form to represent mathematical games is the *strategic form.* Contrary to the extensive form, it does not represent the time dependence of the players' strategy selections and thus shrinks the game into a single time step. A strategic form game is fully determined by the set of players, the strategies available to each player and the rule determining how the payoffs of the players depend on their strategy selections. A compact representation of games in the strategic form is a multidimensional array of numbers corresponding to the joint strategy choices of the players. Therefore, games in the strategic form are usually referred to as *matrix games* and particularly in the case of two players, if the payoff matrices for both players are separated, as *bimatrix games.* In general, an N-person matrix game is defined as follows:

Definition 3.1 *A* matrix game *is a tuple* $\Gamma = (A^1, \ldots, A^N, r^1, \ldots, r^N)$, *where* N *is the number of players,* A^i *is the strategy space for player* i *and* $r^i : A^1 \times A^2 \times \ldots \times A^N \rightarrow \mathbb{R}$ *is the payoff function for player* i.

In addition to the *pure strategies*, $a^i \in A^i$ it is allowed that players can randomize over their pure strategies. Let $\Delta(A^i)$ be the set of all distributions over the strategies of player i.

Then, an arbitrary *mixed strategy*, i.e. a randomization over the pure strategies of player i, is denoted as $\sigma^i \in \Delta(A^i)$.

Usually, as the strategic form games are static, it is assumed that the players *implement* (or are ready to implement and can not change their decisions anymore) their strategy selections simultaneously and get their rewards after the implementation. This often implies that the players also *select* their strategies simultaneously. However, with some solution concepts, e.g. the *Stackelberg* equilibrium, the players have an ordering in their decision process and the payoff values realize only after all players have selected their strategies.

An example matrix game is depicted in Game 3.2. In that game, there are two players, both with three possible strategies. The first number in each cell denotes the payoff value for player 1 and the second number for player 2.

	a^2	b^2	c^2
a^1	$0, 1$	$-2, -1$	$-\frac{3}{2}, \frac{2}{3}$
b^1	$-1, -2$	$-1, 0$	$-3, -1$
c^1	$1, 0$	$-2, -1$	$-2, \frac{1}{2}$

Game 3.2: An example matrix game of two players each having three possible pure strategies.

3.2.4 Correspondence of the extensive form and the strategic form

Extensive form games can be transformed into the strategic form although all references to the timing of the moves in the original extensive form are lost. In many cases, this reduction does not cause any problems and the resulting strategic form games are often much easier to analyze. On the other hand, there exists also games in which the reduction has crucial consequences and the solutions obtained from strategic representations can be quite suspicious.

There are two basic representations of an extensive form game in the strategic form, namely the *normal form* and the *multiagent form*. In the normal form, a strategy of a player defines the player's behavior in all of the player's information states of the game. The normal representation of the extensive form game depicted on the left in Game 3.1 is shown in Game 3.3 and the extensive form game depicted on the right in Game 3.4.

	a^2A^2	a^2B^2	b^2A^2	b^2B^2
a^1	$3, 1$	$3, 1$	$0, 0$	$0, 0$
b^1	$0, 0$	$1, 3$	$0, 0$	$1, 3$

Game 3.3: The normal form representation of the extensive form game depicted in Game 3.1 (left).

If two pure strategies of a strategic form game lead to the same payoff for the player no matter what the player's opponents do, then the strategies are called *payoff equivalent*. Then it is possible to alter the actual game by deleting all but one strategy from each equivalence class of strategies and the resulting game is called *purely reduced normal form*. Moreover, if a

	a^2	b^2
a^1	$3,1$	$0,0$
b^1	$0,0$	$1,3$

Game 3.4: The normal form representation of the extensive form game depicted in Game 3.1 (right).

mixed strategy leads to the same payoff value for the player, regardless of the opponents' play, as a pure strategy, then this pure strategy is said to be *randomly redundant* and can be erased from the game. When the randomly redundant strategies are removed from purely reduced normal form, the game has the same relevant information as the original game but the number of the strategies is smaller. The game is said to be in the *fully reduced normal form.*

In the multiagent representation of an extensive form game, the set of players is modified by defining a player for each information state of the game. A strategy profile in the multiagent representation of an extensive form game is usually referred to as a *behavior strategy profile* and it defines a probability distribution over the strategies available to the player in the corresponding information state. Many mixed strategies may generate an equal behavior (they are *behaviorally equivalent*) in the course of play.

The representations discussed above are very generic and suitable for all extensive form games. If the extensive form game has some special properties or special properties are required for the applied solution concept, e.g. *SubGame Perfectness (SGP)*, a more suitable representation may be found. These issues are discussed at the end of this chapter.

3.3 Solution concepts for games

The goal of game theory is to estimate, given a game in some representation, the possible paths of the play through the game tree in the extensive form. In decision theory, the rational decision maker selects a strategy that maximizes the decision maker's expected utility value. In game theory, the strategy selections of the opponents affect the utility value of the player and therefore the player should take the opponents' behavior into account. In this section, various possible solution concepts for mathematical games in strategic form are introduced. The applicability of a particular solution concept depends crucially on the properties of the game, such as the number of players, the correlation between the players' payoff functions, and the roles of the players in the decision making. In some sense, however, it might me useful to think that all of these solution concepts are only extensions of the maximum operator used in decision theory for multiple decision makers. The solution concept presented in this section are discussed more thoroughly e.g. in Myerson (1991) and in Fudenberg and Levine (1998).

3.3.1 Elimination of dominated strategies

Perhaps the weakest solution concept for solving mathematical games is the elimination of dominated strategies. It is based on the fact that a rational player should never select a strategy that does worse than some other strategy regardless of the opponents' play. It only provides a method to reduce the number of strategies available to the players and does not always lead to a situation in which all players have only one strategy left. The method can be further divided into two subcategories: *reduction of strongly dominated strategies* and *reduction of weakly dominated strategies.*

Reduction of strongly dominated strategies is the weaker one of the elimination methods and it may be possible to continue the reduction further by using the elimination of weakly dominated strategies. The strategy a^i is strongly dominated by the mixed strategy σ^i if the following inequality holds for all strategy profiles $a^1, \ldots, a^{i-1}, a^{i+1}, \ldots, a^N$:

$$r^i(a^1, \ldots, a^{i-1}, a^i, a^{i+i}, \ldots, a^N) < r^i(a^1, \ldots, a^{i-1}, \sigma^i, a^{i+i}, \ldots, a^N), \tag{3.1}$$

where the right hand side is a short-hand notation for the expected payoff value given a mixed strategy σ^i. After a strongly dominated strategy is erased from the set of strategies, other strategies may become strongly dominated and the elimination process can thus be continued. The order in which the strategies are eliminated does not affect the final residual game resulted from the iterated elimination process.

In the game depicted in Game 3.5, the strategy c^2 is dominated for player 2 by the mixed strategy in which player 2 selects strategies a^2 and b^2 with an equal probability. After eliminating c^2, b^1 becomes dominated by a^1 for player 1. In the resulting game, b^2 is then strongly dominated by a^2. This reduction process thus leads to a game in which both players have only one strategy option left and hence the elimination process provides a unique prediction of the game depicted in Game 3.5.

	a^2	b^2	c^2
a^1	$2,3$	$3,0$	$0,1$
b^1	$0,0$	$1,6$	$4,2$

Game 3.5: An example strategic form game. The iterated elimination of strongly dominated strategies leads to the unique prediction of the game, namely the strategy profile $([a^1],[a^2])$.

The other closely related method is the elimination of weakly dominated strategies. The strategy a^i is weakly dominated by the mixed strategy σ^i if the following inequality holds for all strategy profiles $a^1, \ldots, a^{i-1}, a^{i+1}, a^N$:

$$r^i(a^1, \ldots, a^{i-1}, a^i, a^{i+i}, \ldots, a^N) \leq r^i(a^1, \ldots, a^{i-1}, \sigma^i, a^{i+i}, \ldots, a^N) \tag{3.2}$$

with the strict inequality holding for at least one of the opponents' strategies. Note that weakly dominated strategies include all strongly dominated strategies and hence the reduction process often leads to smaller residual games. However, this reduction process is order dependent and therefore the resulting residual game depends on the order in which the weakly dominated strategies are eliminated.

As stated above, the elimination of dominated strategies is not a real solution concept for games in strategic form as the residual game may have more than one strategy left for all players. The well-known game named *Battle of Sexes* (for detailed description of this game and the other popular example games, see e.g. Luce and Raiffa, 1957), depicted in Game 3.4, is an example of games in which it is not possible to eliminate any strategies by using the above discussed elimination processes.

3.3.2 Stackelberg equilibrium

When the Stackelberg solution concept is applied to a matrix game, the players are put in some ordering and all the players are assumed to know and accept this ordering. In the two-player case with player 1 selecting a strategy first, player 1 is called the *leader* and player 2 the *follower*. It is often said that the leader is capable of enforcing or announcing a strategy selection to the follower who then selects, in its turn, its own strategy rationally based on this enforcement. Note that this does not mean that the leader has an advantage over the follower; in fact since the leader has to put its cards on the table first, the follower can enforce the leader to play some strategy. For example, in Game 3.6, if player 1 (the row player) is acting as the leader and player 2 as the follower, the leader has to consider the rational responses of the follower. If the leader selects the strategy a^1, the follower's rational response is a^2. If the leader selects b^1, the follower responds with the strategy b^2 and if the leader selects c^1, the follower selects c^2. Clearly, the leader's strategy choice b^1 leads to the best payoff for the leader and therefore it is also the unique rational strategy selection for the leader.

	a^2	b^2	c^2
a^1	$1,1$	$0,0$	$0,0$
b^1	$0,0$	$3,1$	$0,0$
c^1	$0,0$	$0,0$	$2,1$

Game 3.6: An example strategic form game with the unique Stackelberg equilibrium strategy profile $([b^1],[b^2])$.

The only requirement of the Stackelberg equilibrium is that the follower's response should be unique. If this is not the case, some additional requirements should be set. In Game 3.7, there is an example in which the follower's response is not unique when the leader selects its strategy a^1. However, if the follower always chooses the first of its rational responses, the unique Stackelberg equilibrium for this game is the strategy profile $([a^1],[a^2])$. Another possibility is that the leader acts in a risk-aware manner and always selects the strategy with the minimal possible loss. In this case, the unique equilibrium is the strategy profile $([b^1],[b^2])$.

	a^2	b^2
a^1	$3,1$	$0,1$
b^1	$0,0$	$1,2$

Game 3.7: An example strategic form game in which player 2 does not have the unique rational response when player 1 selects the strategy a^1.

In general, there could be several possible Stackelberg strategies for the leader with the same value. Therefore, there is no need for the leader to fluctuate among its rational strategies. This leads to the fact that the value of the Stackelberg equilibrium point is unique for the leader and can also be unique for the follower, which is a very important property when this equilibrium concept is applied in learning algorithms. This issue is discussed more deeply in the next chapter. Another important property of the Stackelberg equilibrium concept is that it always exists in pure strategies and is therefore very fast to compute (it is also possible to introduce the concept of a mixed strategy Stackelberg solution that leads to a bilevel optimization problem; see Bard (1998)). In addition, in the two-player case, only the leader has to have a model of its opponent, i.e. the follower. That reduces the total amount of space required to compute the solution of the game. If the game has a special payoff structure, i.e. the team or the zero-sum payoff structure, the Stackelberg equilibrium has a special meaning. These two cases are discussed below with the MaxMin and MaxMax solutions.

3.3.3 Correlated equilibrium

When the Stackelberg solution concept is applied to a strategic form game, there is a strict ordering among the decision makers. In the next two solution concepts, i.e. the correlated equilibrium concept and the Nash equilibrium concept, the roles of all players are equal and they make their strategy selections simultaneously. As none of the players reveal their strategy selections, all the players are uncertain about their opponents' strategies and therefore the equilibrium strategies are often mixed.

In the correlated equilibrium concept, it is assumed that in addition to players, there exists a *mediator*, a human or a machine that stochastically recommends pure strategies to the players. Each player may or may not obey this recommendation. The mediator draws its recommendations from the following joint distribution:

$$\delta \in \Delta(A^1 \times \ldots \times A^N). \tag{3.3}$$

The distribution δ is a correlated equilibrium if it satisfies the following *strategic incentive constraints*:

$$\sum_{a^{-i} \in A^{-i}} \delta(a^i, a^{-i})(r^i(a^i, a^{-i}) - r^i(b^i, a^{-i})) \geq 0, \forall i \in N, \forall a^i \in A^i, \forall b^i \in A^i, \tag{3.4}$$

where N is the number of players and $A^{-i} = A^1 \times \ldots \times A^{i-1} \times A^{i+1} \times \ldots \times A^N$. The meaning of these constraints is that it is rational for all players to obey the recommendation of the mediator if the distribution δ satisfies these constraints. The mediator should rank feasible recommendations by using some fitness function, e.g. the sum of the players' payoff values.

	a^2	b^2
a^1	$3,1$	$0,0$
b^1	$0,0$	$1,4$

Game 3.8: An example strategic form game with the correlated equilibrium $([b^1],[b^2])$.

As an example, consider Game 3.8. If the mediator recommends the joint strategy that maximizes the summed utility value of the players, the correlated equilibrium point can be obtained by solving the following linear program:

$$
\begin{array}{l}
\max f = 4\delta(a^1,a^2) + 5\delta(b^1,b^2) \\
\begin{array}{rcrcl}
(3-0)\delta(a^1,a^2) & + & (0-1)\delta(a^1,b^2) & \geq & 0 \\
(0-3)\delta(b^1,a^2) & + & (1-0)\delta(b^1,b^2) & \geq & 0 \\
(1-0)\delta(a^1,a^2) & + & (0-4)\delta(b^1,a^2) & \geq & 0 \\
(0-1)\delta(a^1,b^2) & + & (4-0)\delta(b^1,b^2) & \geq & 0
\end{array} \\
\delta(a^1,a^2), \delta(a^1,b^2), \delta(b^1,a^2), \delta(b^1,b^2) \geq 0 \\
\delta(a^1,a^2) + \delta(a^1,b^2) + \delta(b^1,a^2) + \delta(b^1,b^2) = 1.
\end{array}
$$

The unique solution of this linear program is $\delta(a^1,a^2) = \delta(a^1,b^2) = \delta(b^1,a^2) = 0$ and $\delta(b^1,b^2) = 1.0$, which is also intuitively clear from the example. It is quite efficient to solve a game by using the correlated equilibrium concept and the corresponding linear program. Moreover, due to the existence of the mediator, the equilibrium selection is always coordinated. However, in many problems the players are not willing to communicate with each other or to use a mediator and therefore some other solution concept, such as the Nash equilibrium concept, could be more useful.

3.3.4 Nash equilibrium

Independent mixed strategies $(\sigma^1, \ldots, \sigma^N)$ are said to constitute a Nash equilibrium solution of a game if there is no compulsion to deviate from this equilibrium point for any rational player alone. Therefore the Nash equilibrium concept provides a secure solution concept for the game if all players know that other players will also play the same Nash equilibrium solution. Mathematically, a mixed strategy profile $(\sigma^1, \ldots, \sigma^N)$ constitutes a Nash equilibrium if the following inequality holds for all $\sigma^i \in \Delta(A^i)$ and for all i:

$$r^i(\sigma^1_*, \ldots, \sigma^{i-1}_*, \sigma^i, \sigma^{i+1}_*, \ldots, \sigma^N_*) \leq r^i(\sigma^1_*, \ldots, \sigma^N_*). \tag{3.5}$$

There can be infinitely many Nash equilibria in a finite game. Hence, players should agree on which equilibrium point to select. This agreement can be done for example by allowing some social rules in decision making. Note that if all players calculate the set of possible equilibria and there exist a social rule for the equilibrium point selection, e.g. always select the first equilibrium in the list, it is required that all the players use the same (and deterministic) method for computing the list of equilibria.

As shown by Nash (1950b), every finite game in strategic form possesses at least one Nash equilibrium point in mixed strategies. However, as there does not always exist Nash equilibria in pure strategies, some sophisticated optimization procedure is needed for calculating the Nash equilibria. In principle it is possible to enumerate all pure strategy supports and examine whether there exists a Nash equilibrium. However, the number of supports grows very fast with the size of the game and therefore this method is only feasible with small games.

In two-person games, the *Lemke-Howson algorithm* provides a way to find at least one Nash equilibrium. The method provides a solution to the linear complementarity problem (Cottle, Stone and Pang, 1992) and is globally convergent. However, due to the linearity assumption, the method is not suitable for games with more than two players. Moreover, the computational complexity of the Lemke-Howson algorithm is unknown (the worst case lower bound complexity is, however, known to be exponential; see Murty (1978)).

The Lemke-Howson algorithm can also be used for trying to determine the set of all Nash equilibria. It can compute the set of *accessible* Nash equilibria but this set does not always include all equilibria of the game. When the number of players is more than two, the problem of finding Nash equilibria is considerably harder and it is still an open research question if computationally efficient methods exist for this purpose. A good source of information on calculation of the Nash equilibria is McKelvey and McLennan (1996). An overview of the methods applicable for two-player games can be found in von Stengel (2002). Some complexity results about Nash equilibria can be found in Conitzer and Sandholm (2003c).

In the battle of sexes game depicted in Game 3.4, there are two pure strategy Nash equilibria, in which both players select strategies a, i.e. the strategy profile $([a^1],[a^2])$ or both select b, i.e. $([b^1],[b^2])$. In addition to these pure strategy equilibria, there are also a mixed strategy equilibrium $(0.75 * [a^1] + 0.25 * [b^1], 0.25 * [a^2] + 0.75 * [b^2])$.

In games with special payoff structures. e.g. two person zero-sum games or team games, the Nash equilibrium concept has some special forms. These issues are discussed more thoroughly below.

3.3.5 MaxMin solution

Much of the early work in game theory was on two-person zero-sum games, i.e. games where two players have totally opposite interest and therefore there is the maximal negative correlation between their payoff functions. Mathematically, in two-person zero-sum games it holds for all $a^1 \in A^1$ and for all $a^2 \in A^2$:

$$r^1(a^1,a^2) = -r^2(a^1,a^2). \tag{3.6}$$

The strategy profile $\sigma^i \in \Delta(A^i)$ is a MaxMin strategy for player 1 if and only if:

$$\sigma^1 \in \arg \max_{\tau \in \Delta(A^1)} \min_{a^2 \in A^2} r^1(\tau, a^2). \tag{3.7}$$

In here, the minimum could also be defined over the mixed strategies of player 2. However, since the min operator is “inside” the max operator, it always attains its minimum in pure strategies. At the same time, for player 2 it holds that:

$$\sigma^2 \in \arg \min_{\tau \in \Delta(A^2)} \max_{a^1 \in A^1} r^1(a^1, \tau). \tag{3.8}$$

Note that neither of these equations includes the actual equilibrium strategy of the opponent and therefore it is not required to coordinate the equilibrium selection, although there can be more than one equilibrium in the game.

Because the utility values of player 2 are negations of the values of player 1, the same equations apply also for player 2 when the utility functions are changed from r^1 to r^2. The principal strength of the MaxMin solution is that it provides a security level solution to the game: it is a best response against the worst possible strategy selection of the opponent and the corresponding outcome is the worst case outcome. If the player assumes that it faces a zero-sum game, it can do no worse but possibly better against an unknown opponent.

If it is assumed that both players make their decisions simultaneously, a MaxMin solution is guaranteed to exist only in mixed strategies. However, if the players do their strategy selections sequentially, there exists also a MaxMin solution in pure strategies corresponding to the two-player Stackelberg solution in zero-sum games. However, in this case the solution is dependent, as in the general-sum case, on the ordering of the players.

3.3.6 MaxMax solution and its extensions

The opposite class of games to zero-sum games is games in which the players share the same utility function and hence maximal positive correlation exists between the payoff functions. In this case, the problem reduces, in fact, to the decision problem in which there is a metaplayer corresponding to all the players. The strategies available to the metaplayer are joint strategies of all the players sharing the same utility function. The metaplayer acts rationally and therefore implements a joint strategy maximizing the payoff function over the joint strategies (MaxMax solution).

The main problem with the MaxMax solution is the coordination of strategy selections. The utility function defined over the joint strategies may reach its maximum in several points and therefore all the players should select the same equilibrium point. A solution to this problem is to put players in some order known to all of them and then apply the Stackelberg solution concept (Stackelberg solution corresponds to the maximum over all joint strategies if all players share the same payoff function) to the game.

3.4 Stackelberg solution and normal representation

All strategic form games can be thought to have been derived from the extensive form games. Thus all solution concepts, except for the correlated equilibrium concept, introduced in this chapter are only special cases of Nash equilibria of extensive form games represented in the strategic form. Consider the two extensive form games depicted in Game 3.1. The only difference between these games is the information state of player 2: in the first case player 2 does know the strategy of player 1 and in the second case player 2 does not know what strategy player 1 has deployed at the previous time step. The normal representations of the games are depicted in Games 3.3 and 3.4. However, as player 2 has two separate information sets, there are four pure strategy options available for player 2 in the normal representation of the game. Moreover, there are several Nash equilibria in this normal representation and all of these correspond to Nash equilibria in the original extensive form game. One of these equilibria has the SGP property, i.e. it is a rational solution for each proper subgame in the corresponding extensive form game (roughly speaking, proper subgames are extensive form games originating from some node in the original extensive form games, e.g. the original game and leaf nodes). This solution can be achieved by backward induction starting from the leaf nodes and evaluating the value of each proper subgame. When this procedure is continued backwards, eventually, at the root node, the optimal equilibrium value of the whole game is obtained. In the extensive form game depicted on the left in Game 3.1, this solution can be achieved by representing the game in the normal representation depicted in Game 3.4 and applying the Stackelberg equilibrium concept to this game.

The SGP property and the Stackelberg equilibrium concept can be used for constructing different player hierarchies. Consider the game of three players in which one of the players, say player 1, is acting as the leader and is thus capable of enforcing its strategy to two followers, i.e. players 2 and 3. Then the payoff function for each player is a three dimensional tensor. Now, when the leader fixes its strategy, the followers face a two-player strategic form game, which may be different for each enforcement. The followers are at the same level of the hierarchy and utilize the Nash equilibrium concept in their mutual game. If the leader now knows which Nash equilibrium the followers will select, it can choose a rational Stackelberg strategy. As an example, consider the strategic form games depicted in Games 3.9 and 3.10 corresponding to the leader's strategy choices of a^1 and b^1, respectively. Now, if in Game 3.9 the followers select the pure strategy Nash equilibrium corresponding to the strategy profile $([a^2],[a^3])$, the overall strategy profile is $([a^1],[a^2],[a^3])$ and produces some payoff, say 1, for the leader. Similarly, if the followers select the Nash equilibrium $([a^2],[b^3])$ in Game 3.10, the overall strategy profile is $([b^1],[a^2],[b^3])$ leading to payoff 2 for the leader. Clearly, the leader maximizes its own payoff by selecting the strategy b^1.

	a^3	b^3
a^2	$1,1$	$0,0$
b^2	$0,0$	$1,1$

Game 3.9: The resulting subgame when the leader selects the strategy a^1.

	a^3	b^3
a^2	$0,0$	$1,1$
b^2	$1,1$	$0,0$

Game 3.10: The resulting subgame when the leader selects the strategy b^1.

Chapter 4

Multiagent reinforcement learning in Markov games

A fundamental assumption behind Markov Decision Processes (MDPs) is that the environment of the learning agent satisfies the Markov property, i.e. state transitions may depend only on the current state and the action selection of the agent. In many problems this is not the case; there are additional factors that affect the state transition probabilities. For example, it is possible that the real state of the environment is not known to the agent, or other agents can exist in the environment. In this thesis, the focus is on the multiagent learning problem and how to solve it by modeling the behavior of other agents.

In this chapter, two extensions of MDPs to multiagent domains are discussed. In the first extension, *Alternating Markov Games (AMGs)*, only one agent makes its action selection in each state and the environment changes its state based on this action selection. In the more complex and general model, *Markov Games (MGs)*, all agents implement their actions simultaneously in each state and the environment changes its state based on these actions. In this chapter, only models with two agents are discussed. However, the presented models and methods can easily be extended for an arbitrary number of agents.

4.1 Mathematical principles

Although the AMG model is simpler and closer to the single-agent MDPs than the "real" MG model, it can also be considered as a special case of this more general model. Therefore, the general MG model is discussed first. In an MG, the process changes its state according to the action choices of all agents in the system and can thus be seen as a multicontroller MDP. A thorough discussion about MG model can be found in Filar and Vrieze (1997). Formally, an MG can be defined as follows:

Definition 4.1 *A* Markov game *(*stochastic game*) is defined as a tuple* $(S, A^1, A^2, p, r^1, r^2)$, *$S$ is the set of all states, A^i is the set of all actions for each agent $i \in \{1, 2\}$, $p : S \times A^1 \times A^2 \to \Delta(S)$ is the state transition function, $r^i : S \times A^1 \times A^2 \to \mathbb{R}$ is the reward function for agent i. $\Delta(S)$ is the set of all probability distributions over the set S.*

In each state, both agents select their actions by solving the associated bimatrix game. In this thesis, we make a distinction between two types of action selection:

1. Both agents select their actions simultaneously. In this case, the agents are uncertain about their opponent's current move and the action selection procedure is therefore probabilistic. The use of the Nash or the correlated equilibrium concepts lead to this type of action selection. This case is referred to as the symmetric MG model.
2. Agents select their actions sequentially and the actions are implemented simultaneously. Sequentiality implies that there should be an ordering among the agents and that the action selection procedure is deterministic. The use of Stackelberg equilibrium leads to this type of action selection. This case is referred to as the asymmetric MG model. Asymmetric MGs can also be represented as enlarged AMGs. This correspondence is discussed later in this chapter.

As in single-agent reinforcement learning (RL), the behavior of an agent is specified by a policy function. In an MG, there are separate policy functions for each agent and the goal of the agent is to find an equilibrium policy, i.e. a policy that is a best response to its opponent's policy. In the symmetric case, the policy function is defined as follows:

$$\pi^i : S \to A^i. \tag{4.1}$$

Note that the action selection procedure in the symmetric case is actually probabilistic. However, the randomization is assumed to be "inside" the policy function and therefore the policy is expressed as a function from the state space to the agent's action space. In the asymmetric case, the action selection procedure is deterministic. If agent 1 selects its action prior to agent 2 (agent 1 is acting as the leader and agent 2 as the follower), agent 1 should model both its own and also its opponent's behavior. Therefore, the policy function of agent 1 is of the form of Eq. (4.1). Agent 2 does not have a model of agent 1 and is thus unable to make its action choice alone. Indeed, agent 1 announces (enforces) its action to agent 2 and agent 2 responds rationally to this announcement. Therefore, the policy function of agent 2 depends on the current state and the current enforcement, i.e.:

$$\pi^2 : S \times A^1 \to A^2. \tag{4.2}$$

In the case of two agents, an equilibrium policy is defined as follows:

Definition 4.2 *Let Π^1 and Π^2 be the policy spaces for agents 1 and 2, respectively. Policies π^1_* and π^2_* constitute equilibrium policies of an MG if the following inequalities hold for all $\pi^1 \in \Pi^1$ and $\pi^2 \in \Pi^2$ in each state s:*

$$V^1_{\pi^1,\pi^2_*}(s) \leq V^1_{\pi^1_*,\pi^2_*}(s)$$
$$V^2_{\pi^1_*,\pi^2}(s) \leq V^2_{\pi^1_*,\pi^2_*}(s).$$

If agents 1 and 2 follow the policies π^1 and π^2, respectively, the expected discounted utility R^i of agent i is the following:

$$V^i_{\pi^1,\pi^2}(s) = E_{\pi^1,\pi^2}[R^i|s_0 = s] = E_{\pi^1,\pi^2}\left[\sum_{t=0}^{\infty}\gamma^t r^i_{t+1}|s_0 = s\right], \tag{4.3}$$

where r^i_{t+1} is the immediate reward associated to the state transition after the action selections in state s_t for agent i. γ is a discount factor. Moreover, the value for each state-actions tuple is:

$$\begin{aligned} Q^i_{\pi^1,\pi^2}(s, a^1, a^2) &= E_{\pi^1,\pi^2}[R^i|s_0 = s, a^1_0 = a^1, a^2_0 = a^2] \\ &= r^i(s, a^1, a^2) + \gamma \sum_{s' \in S} p(s'|s, a^1, a^2)V^i_{\pi^1,\pi^2}(s'). \end{aligned} \tag{4.4}$$

Note that this equation for the Q-value is a very natural extension to the single-agent case. The main difference with the single-agent case is that in the multiagent case, a matrix game is associated with each state $s \in S$ whereas in the single-agent case these matrix games reduce to vectors containing the utility values for the learning agent. For the equilibrium policies π^1_* and π^2_* it should hold in each state that:

$$V^i_{\pi^1_*,\pi^2_*}(s) = \mathrm{Val}\{Q^i_{\pi^1_*,\pi^2_*}(s)\}, \tag{4.5}$$

where $\mathrm{Val}\{\cdot\}$ is one of the solution concepts discussed in Chapter 3 and $Q^i_{\pi^1_*,\pi^2_*}(s)$ is the matrix game associated with the state s. Note that this value is not necessarily unique. For example, in the case of Nash equilibrium concept, there can be several equilibria with different values.

A thorough discussion about the numeric solution methods for MGs can be found in Filar and Vrieze (1997). As mentioned in Chapter 2, the concept of the MDP was originally proposed by Bellman (1957b) as a discrete version of the optimal control problem. Multicontroller version of the optimal control problem is discussed by Basar and Olsder (1982), which also includes references to the stochastic discrete multicontroller problem, i.e. stochastic game.

4.2 AMGs

In this section, the general MG model is simplified so that the state space is divided into two distinct subspaces $S^1 \in S$ and $S^2 \in S$ so that $S^1 \cap S^2 = \emptyset$ and $S^1 \cup S^2 = S$. In states S^1, agent 1 makes its action selection and the system changes its state according to this selection. Correspondingly, in states S^2, the state transitions are controlled by agent 2. As there is now only one decision maker in each state, the model is very close to the single-agent reinforcement learning model and equilibrium policies are always deterministic.

In AMGs, for agent 1 (equations are symmetric for agent 2) when arbitrary policies π^1 and π^2 are followed it holds in a state $s^1 \in S^1$ that:

$$Q^1_{\pi^1,\pi^2}(s^1, a^1) = r^1(s^1, a^1) + \gamma \sum_{s' \in S} p(s'|s^1, a^1)V^1_{\pi^1,\pi^2}(s') \tag{4.6}$$

and in a state $s^2 \in S^2$:

$$Q^1_{\pi^1,\pi^2}(s^2,a^2) = r^1(s^2,a^2) + \gamma \sum_{s' \in S} p(s'|s^2,a^2) V^1_{\pi^1,\pi^2}(s'). \tag{4.7}$$

In the above equations, a^i is an action selection of agent i. The value of the state s, $V^1_{\pi^1,\pi^2}(s)$, is defined as follows:

$$V^1_{\pi^1,\pi^2}(s) = \begin{cases} Q^1_{\pi^1,\pi^2}(s,\pi^1(s)) & s \in S^1 \\ Q^1_{\pi^1,\pi^2}(s,\pi^2(s)) & s \in S^2. \end{cases} \tag{4.8}$$

For equilibrium policies π^1_* and π^2_* it holds that:

$$V^1_{\pi^1_*,\pi^2_*}(s) = \begin{cases} \max_{b \in A^1} Q^1_{\pi^1_*,\pi^2_*}(s,b) & s \in S^1 \\ Q^1_{\pi^1_*,\pi^2_*}(s,\pi^2_*(s)) & s \in S^2, \end{cases} \tag{4.9}$$

meaning that the agent should behave optimally in all states in which it controls the process. π^2_* is defined in the natural way:

$$\pi^2_*(s) = \arg\max_{b \in A^2} Q^2_{\pi^1_*,\pi^2_*}(s,b). \tag{4.10}$$

In addition, if the interests of the agents are fully conflicting, the AMG has a zero-sum payoff structure. In this case the whole system can be modeled by using a single Q-function and the agents can be isolated from each other. In this case, Eq. (4.6) for optimal policies π^1_* and π^2_* reduces to the following form:

$$Q_{\pi^1_*,\pi^2_*}(s^1,a^1) = r(s^1,a^1) + \gamma \sum_{s' \in S} p(s'|s^1,a^1) V_{\pi^1_*,\pi^2_*}(s') \tag{4.11}$$

and the value $V_{\pi^1_*,\pi^2_*}(s)$ in a state s is:

$$V_{\pi^1_*,\pi^2_*}(s) = \begin{cases} \max_{b \in A^1} Q_{\pi^1_*,\pi^2_*}(s,b) & s \in S^1 \\ \min_{b \in A^2} Q_{\pi^1_*,\pi^2_*}(s,b) & s \in S^2. \end{cases} \tag{4.12}$$

Applying reinforcement learning to game playing is one of the most studied domains of reinforcement learning in the literature. In most of the traditional board games, the players take turns during the game and therefore it is natural to formalize these games as AMGs. Various RL based methods have been applied to game playing tasks, e.g. checkers (Samuel, 1959), chess (Thrun, 1994), go (Schraudolph, Dayan and Sejnowski, 1993) and backgammon (Tesauro, 1992; Boyan, 1992).

4.3 Correspondence between models

Both extensive form games discussed in Chapter 3 and MGs are dynamic models of decision making, i.e. the time dependence is a crucial feature in both models. A close relationship exists between these two models; in fact it is possible to model and solve some problems using either one of these models. On the other hand, AMGs are special cases of the more general MGs. Perhaps surprisingly, asymmetric MGs can also be represented as enlarged AMGs. These relationships are discussed in this section.

4.3.1 Correspondence between MGs and extensive form games

Every state in an MG contains a normal representation of the extensive form game, the exact form of which depends on the applied solution concept. For example, in a two-agent symmetric MG, the decision nodes of agent 2 are in the same information state. Correspondingly, in the case of the Stackelberg equilibrium concept with agent 1 acting as the leader, both decision nodes of agent 2 are in distinct information states. As an example, consider a two-agent symmetric MG with two states with both agents having two actions available in each state and with deterministic state transitions from a state to another (the same transition for each joint action). If the process has infinite horizon, i.e. there are no time limits set, the system can be modeled as the extensive form game illustrated in Fig. 4.1. Note that the game is indeed infinite as there are no leaf nodes in the game tree. Therefore, it is not possible to model infinite horizon problems as extensive form games. In addition, even with finite horizon problems, the extensive form game grows very fast with the length of the time horizon and therefore it is not tractable to model these problems as extensive form games either.

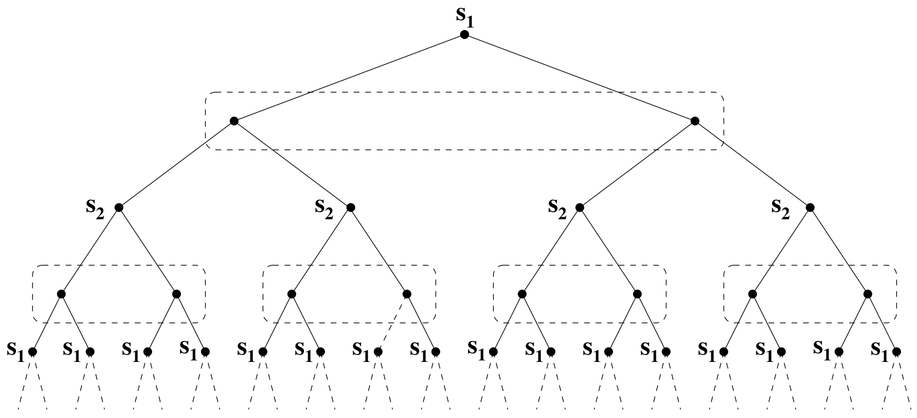


Figure 4.1: The correspondence between MGs and extensive form games.

4.3.2 Correspondence between AMGs and asymmetric MGs

An asymmetric MG can also be represented as an enlarged AMG, i.e. the AMG that has more states than the original asymmetric MG. In fact, a general MG model endowed with the Stackelberg equilibrium concept constitutes a special case of AMG model in which the system changes its state stochastically only when both agents have selected their actions.

As an example of the correspondence between AMGs and asymmetric MGs, consider a simple MG including only one state and two agents with two possible options. The state can thus be represented with two 2 x 2 matrices. Further, if agent 1 is acting as the leader and agent 2 as the follower, the optimal policy can be determined simply by solving the bimatrix game using the Stackelberg solution concept. In the corresponding enlarged AMG, the original state is divided into three substates, namely s_1, s_2 and s_3. The state s_1 is controlled by agent 1 and states s_2 and s_3 by agent 2. The state transitions are deterministic and can

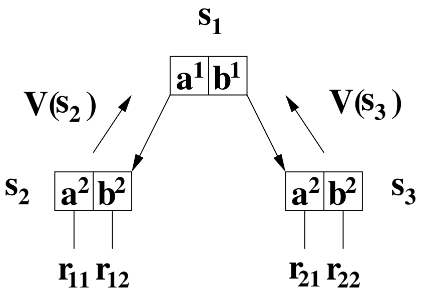


Figure 4.2: A state in an MG endowed with the asymmetric equilibrium concept described as an AMG. The state s_1 is controlled by agent 1 and the states s_2 and s_3 by agent 2. In the states s_2 and s_3, rewards are for agent 1.

occur from s_1 to s_2 or to s_3. Both agents get their rewards when agent 2 has done its action selection and thus all Q-values in the states s_2 and s_3 are determined by these rewards. On the other hand, there are no rewards when the system changes its state from s_1 to either s_2 or s_3 and therefore the Q-values in the state are determined by the values of these two states and the behavior of agent 2. Clearly, agent 1 always chooses the action with the highest utility value in the state s_1. This setting is illustrated in Fig. 4.2. Note the correspondence with Fig. 4.2 and the game on the left in Fig. 3.1 in Chapter 3.

As was seen in the above example, MGs endowed with the asymmetric solution concept can be represented as AMGs. However, MGs are usually more compact representations for these problems. The reason for this is that AMGs are more general in the way that they allow stochastic transitions and rewards after all state transitions, which never happens in asymmetric MGs. Moreover, the number of states is larger when a problem is modeled with AMGs and this may hinder the interpretation of the model.

4.4 RL in multiagent settings

Reinforcement learning methods for both AMGs and general MGs are straightforward extensions of single-agent learning rules. However, opposite to single-agent learning, the convergence of these methods can only be proved in some special cases. The convergence properties of RL methods in MGs as well as in AMGs are discussed e.g. in Littman (1996), Szepesvári and Littman (1999) and in Publication I.

4.4.1 RL in AMGs

The Q-learning rule in AMGs is almost identical to the single-agent update rule. For agent 1 in state $s^1 \in S^1$, it takes the following form (for agent 2, the rules are symmetric):

$$Q^1_{t+1}(s^1_t, a^1_t) = (1-\alpha_t)Q^1_t(s^1_t, a^1_t) + \alpha_t[r^1_{t+1} + \gamma V^1_t(s_{t+1})], \tag{4.13}$$

where

$$V_t^1(s_{t+1}) = \begin{cases} \max_{b \in A^1} Q_t^1(s_{t+1}, b) & s_{t+1} \in S^1 \\ Q_t^1(s_{t+1}, z) & s_{t+1} \in S^2. \end{cases} \tag{4.14}$$

In Eq. (4.14), z is the rational action choice of agent 2:

$$z = \arg\max_{b \in A^2} Q_t^2(s_{t+1}, b). \tag{4.15}$$

In the zero-sum case, the system can be modeled using only one Q-function and there is no need for an explicit model of the opponent. The convergence of the zero-sum AMGs was proved by Littman (1996). For the general-sum case, convergence is not guaranteed.

4.4.2 Symmetric learning in MGs

The Q-values of a symmetric MG can be learned using similar methods as in single-agent reinforcement learning problems. In this thesis, only Q-learning-type off-policy methods are discussed. The general off-policy learning rule takes the following form:

$$Q_{t+1}^i(s_t, a_t^1, a_t^2) = (1 - \alpha_t) Q_t^i(s_t, a_t^1, a_t^2) + \alpha_t [r_{t+1}^i + \gamma \mathrm{Val}\{Q_t^i(s_{t+1}\}], \tag{4.16}$$

where Val$\{\cdot\}$ is one of the solution concepts discussed in Chapter 3. Its purpose is to evaluate the value of the matrix game associated with state s_{t+1}. The choice of this solution concept is dependent on the type of the problem at hand and it has a crucial effect on convergence properties of the above update rule. Some symmetric solution concepts are listed below with possible application areas and their major properties. An extensive survey on the learning methods in symmetric MGs is Littman (2001b).

MaxMax. This solution concept is a natural choice in team MGs, in which all agents share the same utility function. The operator simply returns the maximal value of the matrix game and therefore the learned optimal policy is always deterministic. The above update rule converges with probability one (Littman, 2001b) when the MaxMax operator is used. However, the convergence does not imply that the agents also learn to play optimally. The value function may reach its maximum value in several points and all the agents should select the same point; otherwise the resulting policy is not an equilibrium policy. It is possible to solve this problem by setting an ordering among the agents, i.e. to apply the asymmetric solution concept in each state or to use some other methods for learning to play optimally (see e.g. Wang and Sandholm, 2003). RL in a team of learning agents was applied to elevator control in (Crites, 1996).

MaxMin. This solution concept is suitable for problems with two learning agents in which the learners' interests are fully conflicting, i.e. the payoff structure of the problem is zero-sum. The solution is stochastic in general and can be calculated using a simple linear program. With this solution concept, the above update rule converges with probability one (Littman, 2001b). Note that it is possible to use this concept also with general-sum games. In this case, it finds the worst-case solution assuming that the opponent is always minimizing the agent's utility value. An off-policy method utilizing the MaxMin solution concept was originally proposed by Littman (1994). This method is also the first one utilizing RL in MGs and the method was later extended, combining the MaxMax and MaxMin concepts, to general-sum problems in Littman (2001a). Reinforcement learning applied to differential games modeled as zero-sum MGs with large state and action spaces is discussed in Sheppard (1997). A more recent study of RL in zero-sum problems is Banerjee, Sen and Peng (2001), in which on-policy RL methods are applied to these problems.

Nash. The Nash solution concept is a generalization of the MaxMin concept to general-sum games with an arbitrary number of agents. In general, the optimal policy is stochastic. With general-sum problems, it is not possible to provide convergence proofs for the above update equation. Moreover, in the general case, the computational requirements of the equilibrium point calculation can be high. An off-policy multiagent reinforcement learning method utilizing the Nash solution concept was originally proposed by Hu and Wellman (1998) (see also Hu, 1999; Hu and Wellman, 2003) and the convergence properties of the proposed algorithm are clarified in Bowling (2000). Experimental results of using the Nash equilibrium concept in multiagent reinforcement learning can be found in Hu and Wellman (2000). A slightly modified version of the method introduced in Hu and Wellman (1998) takes into account the possibility that the opponent agent uses a non-equilibrium but stationary policy, see (Suematsu and Hayashi, 2002).

Correlated equilibrium. A Nash equilibrium point consists of independent probability distributions over the action spaces of the agents. A correlated equilibrium is, on the contrary, a joint distribution over the joint strategy space of the players and it can be calculated by using a simple linear program. Also in this case, the convergence of the above update rule is not guaranteed. Applying correlated equilibria in multiagent RL was originally proposed by Greenwald and Hall (2003).

4.4.3 Asymmetric learning in MGs

The general update rule for asymmetric learning is the same as in the symmetric case. However, now the value Val$\{\cdot\}$ is different for the leader and for the follower. The update rule for the leader (agent 1) is as follows:

$$Q^1_{t+1}(s_t, a^1_t, a^2_t) = (1-\alpha_t)Q^1_t(s_t, a^1_t, a^2_t) + \alpha_t[r^1_{t+1} + \gamma \max_{b \in A^1} Q^1_t(s_{t+1}, b, Tb)], \tag{4.17}$$

where the operator $T : A^1 \rightarrow A^2$ conducts the follower's best response to the leader's action enforcement. Similarly, the update rule for the follower is as follows:

$$Q^2_{t+1}(s_t, a^1_t, a^2_t) = (1-\alpha_t)Q^2_t(s_t, a^1_t, a^2_t) + \alpha_t[r^2_{t+1} + \gamma \max_{b \in A^2} Q^2_t(s_{t+1}, g(s_{t+1}), b)], \tag{4.18}$$

where $g(s_{t+1})$ is the leader's action enforcement in state s_{t+1}.

As the two Q-learning processes presented in Eqs. (4.17) and (4.18) are very strongly coupled by enforcements and responses, there are two possible ways to implement the learning system:

1. The leader commits to giving its action enforcements in all states of the system. In this case, only the leader should have a model of the follower available. The follower does not have to have a model of the leader.
2. Both agents have models of their opponents available and they accept their roles. If the leader's enforcement is unique (by a social convention etc.), both agents are capable of calculating the equilibria alone and communication between agents is not needed. This makes it possible to teach the agents individually; the model of the opponent is only an aid in the learning process.

If the leader's enforcement is unique, the value of the equilibrium point is also unique. Thus, there is no need for special equilibrium selection methods in the learning methods utilizing the Stackelberg equilibrium concept. This is an advantage of asymmetric MGs compared to symmetric MGs. Moreover, it is possible to concatenate the symmetric and asymmetric MGs by utilizing the subgame perfectness property as was discussed at the end of the previous chapter.

RL methods for asymmetric MGs were originally proposed by the author in Könönen (2003a). A broader discussion and coverage of the subject can be found in Publication I. The properties of RL in asymmetric MGs are studied further in all publications included in this thesis and comparisons with the symmetric MGs are carried out in Publications I, II and III.

Applying the Stackelberg solution concept for solving stagegames in MGs can be seen as a simple form of *recursive opponent modeling*: the agent utilizes some model of its opponent for making optimal decisions. Recursive opponent modeling has been studied in computer science literature in many sources (see e.g. Gmytrasiewicz and Durfee, 1995; Gmytrasiewicz and Durfee, 2000; Carmel and Markovitch, 1996). However, it has not been applied previously, to the best of the author's knowledge, to evaluating stagegame values in RL algorithms.

Based on the previous work of Gmytrasiewicz and Durfee, different levels of agent modeling in computational market environments were studied in Hu and Wellman (2002). Moreover, Littman and Stone (2001) applied the Stackelberg solution concept against a learning opponent endowed with the single-agent Q-learning algorithm in simple strategic form games. Another related method utilizing learning automata for opponent modeling was discussed by Carmel and Markovitch (1998).

An asymmetric MG forms an infinite extensive form game with perfect information, i.e. all agents know what actions their opponents have previously selected. RL in finite extensive form games has been studied in Huang and Sycara (2003) and Sun and Qi (2000). In the former study, an extensive form game is played several times and learners apply a simple learning automaton and a Q-learning algorithm for estimating their outcomes in the game. In the latter study, an opponent modeling method is applied to certain restricted AMGs.

4.5 Some considerations on numeric approximation

Solving problems with large state-actions spaces leads to the need for using function approximators, such as neural networks. Numeric approximation of value functions in multiagent reinforcement learning methods based on MGs coincides to a great extent with single-agent methods. However, taking the behavior of the opponent into account increases the complexity of the problem. In this section, a brief discussion of numeric methods is given, emphasizing the differences between single-agent and multiagent methods.

In the value-function-based methods, the value-functions or Q-functions are approximated with function approximators. In multiagent settings, there is a different approximator for each agent and the agent tries to optimize the weights of this approximator taking into account the action choices of its opponent. In policy gradient methods, the policy function π is parameterized directly and it forms a joint distribution over the joint action space of the agents. In this thesis, policy gradient methods are only discussed in the case of team MGs, i.e. MGs in which all the agents share the same value function.

4.5.1 Value-function-based methods

In multiagent settings, it is essential to take into account the behavior of the opponent. Therefore, the agent should have a model of its opponent available during learning. In this thesis, it is assumed that the agents observe action selections and rewards of both agents in the system, making it possible to estimate the Q-function of the opponent. The observed states, actions and rewards until the time instance t are collected into the history h_t:

$$h_t = \{s_0, a_0^1, a_0^2, r_1^1, r_1^2, \dots, s_{t-1}, a_{t-1}^1, a_{t-1}^2, r_t^1, r_t^2\}, \tag{4.19}$$

where s_t, a_t^i and r_t^i are the states, actions and rewards, respectively, associated with the time instance t. The superscripts refer to the agent number, i.e. $i = 1, 2$. In some problems, a single history containing all data from the interaction between the agents and the environment may exist. In episodic learning, on the other hand, the history contains all data of the current episode and when the end state is reached, the history is cleared.

In approximator training, the first step is to select a suitable error function and then find such parameters $\boldsymbol{\omega}^i$ that minimize the error function. One possibility is to minimize the expected squared error function:

$$e^i(h_t) = \frac{1}{2} \sum_{s_t \in S} P(s_t|s_{t-1}, a_{t-1}^1, a_{t-1}^2)[r_t^i + \gamma \mathrm{Val}\{Q_{t-1}^i(s_t; \boldsymbol{\omega}^i)\} - Q_{t-1}^i(s_{t-1}, a_{t-1}^1, a_{t-1}^2; \boldsymbol{\omega}^i)]^2, \tag{4.20}$$

where Val$\{\cdot\}$ is one of the solution concepts discussed in Chapter 3 and it generally depends on the weights $\boldsymbol{\omega}^1$ and $\boldsymbol{\omega}^2$. If only one sample of state transitions is observed at each time step, the weights can be updated using the following update rule:

$$\Delta\omega \propto [r_t^i + \gamma \mathrm{Val}\{Q_{t-1}^i(s_t; \boldsymbol{\omega}^i)\} - Q_{t-1}^i(s_{t-1}, a_{t-1}^1, a_{t-1}^2; \boldsymbol{\omega}^i)] \frac{\partial Q_{t-1}^i(s_{t-1}, a_{t-1}^1, a_{t-1}^2; \boldsymbol{\omega}^i)}{\partial \omega}, \tag{4.21}$$

where ω is an arbitrary parameter in $\boldsymbol{\omega}^i$. This rule can be used in an on-line manner after every state transition, or in a batch manner for each completed episode history h_t. However, the above equation is a special case of the Widrow-Hoff learning rule and thus does not take into account the fact that the value estimate of the next state is also a function of the weights $\boldsymbol{\omega}^1$ and $\boldsymbol{\omega}^2$. Therefore Baird (1995) proposed a method that extends Eq. (4.21) to perform true gradient descent of the error function proposed in Eq. (4.20). The multiagent version of the update equation takes the following form:

$$\Delta\omega \propto [r_t^i + \gamma \mathrm{Val}\{Q_{t-1}^i(s_t;\boldsymbol{\omega}^i)\} - Q_{t-1}^i(s_{t-1},a_{t-1}^1,a_{t-1}^2;\boldsymbol{\omega}^i)][\phi\gamma\frac{\partial \mathrm{Val}\{Q_{t-1}^i(s_t;\boldsymbol{\omega}^i)\}}{\partial\omega} \\ - \frac{\partial Q_{t-1}^i(s_{t-1},a_{t-1}^1,a_{t-1}^2;\boldsymbol{\omega}^i)}{\partial\omega}], \tag{4.22}$$

where ϕ is a parameter that controls the direction of the gradient. If $\phi = 0$, Eq. (4.22) reduces back to Eq. (4.21) and in the case of $\phi = 1$, Eq. (4.22) is the real gradient of Eq. (4.20). However, in this case the learning process could be extremely slow due to the fact that the parameters are updated to different directions by two gradient terms in (4.22). Therefore, a combination of the two methods with the value of ϕ between the two extremes both guarantees the convergence of the method and leads to quicker convergence than the real gradient. Some theoretical limits for ϕ are derived in Baird (1995), but usually the value of ϕ can be selected heuristically; it is even possible to change the value during the learning process. Another drawback of Eq. (4.22) is that the operator $\mathrm{Val}\{\cdot\}$ is not in general differentiable with respect to ω in $\boldsymbol{\omega}^i$. For some operators, e.g. maximum, this problem can be circumvented. However, in most cases, it is enough to assume that the nondifferentiable points of $\mathrm{Val}\{\cdot\}$ constitute a zero-volume set and differentiate the corresponding Q-value estimates with respect to an arbitrary parameter (Bertsekas and Tsitsiklis, 1996).

Another approach to value function approximation is to minimize the total error E^i generated by all histories:

$$E^i = \sum_{t=0}^{\infty}\sum_{h_t\in H_t} P(h_t)e^i(h_t), \tag{4.23}$$

where H_t is the space containing all histories of length t and $e^i(h_t)$ is the immediate error function for agent i, e.g. the one presented in Eq. (4.20). In this equation, both $e^i(h_t)$ and $P(h_t)$ are functions of $\boldsymbol{\omega}^1$ and $\boldsymbol{\omega}^2$ and thus differentiating Eq. (4.23) with respect to an arbitrary parameter ω in $\boldsymbol{\omega}^i$ leads to the following equation:

$$\frac{\partial E^i}{\partial\omega} = \sum_{t=0}^{\infty}\sum_{h_t\in H_t} P(h_t)\Bigg[\frac{\partial e^i(h_t)}{\partial\omega} \\ + e^i(h_t)\sum_{j=0}^{t-1}\frac{\partial}{\partial\omega}\Bigg(\ln P(a_j^1|s_j;\boldsymbol{\omega}^1,\boldsymbol{\omega}^2) + \ln P(a_j^2|s_j;\boldsymbol{\omega}^1,\boldsymbol{\omega}^2)\Bigg)\Bigg], \tag{4.24}$$

where $P(a_j^i|s_j;\boldsymbol{\omega}^1,\boldsymbol{\omega}^2)$ is the probability that agent i selects the action a_j^i in a state s_j at the time instance j. This probability term defines the exploration function during the learning. A favorable but not necessary property of the exploration policy is that it will approach the true equilibrium in the limit, i.e. the greedy action selection in single-agent settings. In multiagent settings, a natural way is to calculate an equilibrium point and then deviate from this point with some small probability ϵ. During learning, $\epsilon \to 0$ and thus the action selection probabilities approach the true equilibrium probabilities. Another problem is that in multiagent settings, the exploration probability functions are usually not smooth functions with respect to the parameters $\boldsymbol{\omega}^i$. However, by setting certain simplifying assumptions, as was done in Publication II, this problem can be circumvented and the results can still be good.

An interesting property of the gradient in Eq. (4.24) is that the immediate error function exists directly in the gradient. This property makes it possible to combine value function based methods and direct policy gradients in the same equation. That is particularly useful in non-Markov domains, e.g. in partially observable MDPs (POMDPs), in which normal value-function-based methods often give poor results. Applying Eq. (4.23)-type error functions in single-agent reinforcement learning was originally proposed by Baird and Moore (1999). The resulting methods are sometimes referred to as the VAPS framework, standing for *Value And Policy Search*. The VAPS framework was extended to multiagent systems by the author in Könönen (2003b) and in Publication II. The framework can be initialized in different ways leading to various actual RL methods. If only policy search is utilized, the method corresponds to the REINFORCE algorithm proposed by Williams (1992) (see also Williams, 1988; Weaver and Tao, 2001).

4.5.2 Policy gradient methods

Another approach to the multiagent reinforcement learning problem is to parameterize the policy function directly. A suitable way to describe multiagent systems is to define the policy function (joint policy function) to be a joint distribution over the joint action space in each state:

$$\pi(s, a^1, a^2; \boldsymbol{\theta}) = P(a^1, a^2 | s; \boldsymbol{\theta}), \tag{4.25}$$

where $\boldsymbol{\theta}$ is an arbitrary parameter vector. The distribution defines the probability of selecting the joint action (a^1, a^2) in a state $s \in S$. Note that this approach has a close relationship to the correlated equilibrium concept, where the mediator draws its recommendations from the joint distribution π.

In this thesis, the policy gradient method is applied only to team problems, in which the utility function is the same for each agent. Additionally, the focus is on the start state formulation of the problem, where the agents start from an initial state and try to maximize their long-time utility value.

The object function of the policy gradient method is the expected utility in a start state s_0:

$$\rho(s_0, \pi) = V_\pi(s_0) = \sum_{b \in A^1} \sum_{c \in A^2} \pi(s_0, b, c; \boldsymbol{\theta}) Q_\pi(s_0, b, c). \tag{4.26}$$

By differentiating Eq. (4.26) with respect to an arbitrary parameter θ and substituting the Q-function with an approximation f, parameterized with $\boldsymbol{\omega}$, the policy gradient takes the following form :

$$\frac{\partial \rho}{\partial \theta} = \sum_{s \in S} d_\pi(s) \sum_{b \in A^1} \sum_{c \in A^2} \frac{\partial \pi(s, b, c; \boldsymbol{\theta})}{\partial \theta} f(s, b, c; \boldsymbol{\omega}), \tag{4.27}$$

where $d_\pi(s)$ is the discounted weight (probability) of reaching state s starting from the initial state s_0 and following π. It is a real number (a probability) and therefore it can be excluded from Eq. (4.27) and still get the unbiased estimate of the gradient if the state transitions are sampled by following the joint policy function π.

If the function approximator f fulfills the following compatibility condition

$$\frac{\partial f(s, a^1, a^2; \boldsymbol{\omega})}{\partial \omega} = \frac{\partial \ln \pi(s, a^1, a^2; \boldsymbol{\theta})}{\partial \theta}, \tag{4.28}$$

it can be shown that the error due to the use of the function f in place of Q is orthogonal to the gradient of the policy function π and hence it is possible to use f in place of Q in Eq. (4.27).

Note that by virtue of Eq. (4.28), the selection of the parametric joint policy function also fixes the form of f. For example, in the case of Gibbs distribution, the joint policy function takes the following form:

$$\pi(s, a^1, a^2, \boldsymbol{\theta}) = \frac{e^{\boldsymbol{\theta}^T \phi(s,a^1,a^2)}}{\sum_{b \in A^1} \sum_{c \in A^2} e^{\boldsymbol{\theta}^T \phi(s,b,c)}}, \tag{4.29}$$

where $\phi(s, a^1, a^2)$ is a unit vector with the element corresponding to the state-actions tuple (s, a^1, a^2) set to one. Correspondingly, the compatible function approximator f takes the linear form:

$$f(s, a^1, a^2, \boldsymbol{\omega}) = \boldsymbol{\omega}^T [\phi(s, a^1, a^2) - \sum_{b \in A^1} \sum_{c \in A^2} \phi(s, b, c) \pi(s, b, c; \boldsymbol{\theta})]. \tag{4.30}$$

The parameters $\boldsymbol{\omega}$ can be learned by minimizing some error function, e.g. the one presented in Eq. (4.20). The relationship between the parametrized policy function and the value function approximation were proposed simultaneously by Konda and Tsitsiklis (2000) and Sutton, McAllester, Singh and Mansour (2000) for single-agent problems. The policy gradient method applied to team problems and its extensions to general-sum problems are discussed in Publication IV and in Könönen (2003c). The method proposed by Sutton et al. (2000) was used to game playing in Bowling and Veloso (2002b).

4.5.3 Learning to play optimally

Multiagent RL in MGs involves two major and partially interrelated problems:

- Learning the game structure, or the relevant parts of the game structure, of the problem.
- Learning to play well when the game structure is already known.

If the game structure is already known, e.g. learned by using some of the methods presented above, there can be multiple equilibrium polices, only some of which are optimal. Even in team games, there can be various Nash equilibria with different values and the agents need to be able to coordinate their action selections. A basic text, written from the game theory point of view, on learning to play in games in provided by Fudenberg and Levine (1998). In this section, a short literature survey on this issue from the computer science point of view is provided.

In team MGs, the optimal value of each stagegame is unique and therefore it is enough to use the MaxMax solution concept in the off-policy learning rule to learn the exact game structure. When the agents are playing according to the learned game structure, they should be able to coordinate their actions; otherwise it is possible that the agents select joint actions that do not constitute an equilibrium at all. A straightforward way to circumvent this problem is to set an ordering among the agents. This method was discussed in Boutilier (1996) (for a game theoretical justification of this, see Publication IV). In symmetric team MGs, the method that learns to play an optimal Nash equilibrium was proposed by Wang and Sandholm (2003). Conitzer and Sandholm (2003b) proposed a method for zero-sum games that guarantee that the learner does not lose more than a given amount. The work of Conitzer and Sandholm is consistent with the Win or Learn Fast (WoLF) principle proposed by Bowling and Veloso (2002a). According to this principle, the agent should not to accrue losses (win) or to learn to prevent future losses (learn fast). Conitzer and Sandholm (2003a) also proposed a very general convergent method for repeated games and a method for learning near-Pareto-optimal conventions in repeated games was proposed by Wang and Sandholm (2004). Learning to play a Pareto-optimal Nash equilibrium in strategic form games by allowing some communication between agents was proposed by Verbeeck, Nowé, Lenaerts and Parent (2002).

One of the first attempts to utilize single-agent Q-learning against stationary and learning opponents was carried out by Sandholm and Crites (1995) with different state descriptions and exploration policies. A classical comparison of single-agent Q-learning and multiagent Q-learning (called *joint-action learning*) was carried out by Claus and Boutilier (1998) in fully cooperative repeated games. They utilized a simple (one state) version of the standard Q-learning algorithm in their work. A more advanced method for applying single-agent RL in cooperative multiagent domains was proposed by Lauer and Riedmiller (2000). When the single-agent RL methods are utilized in multiagent domains, the applied exploration policy has a crucial effect for the performance of the learning method. This problem is investigated in detail in Kapetanakis and Kudenko (2002b), Kapetanakis and Kudenko (2002a) and Kapetanakis, Kudenko and Strens (2003) in team games. Convergence properties of simple gradient ascent algorithms in small general-sum games were studied by Singh, Kearns and Mansour (2000). A totally different approach for fitting single-agent RL methods to multiagent domains is the *COllective INtelligence (COIN)* framework (see e.g. Wolpert and Tumer, 2000). The basic principle of COIN is to define local reward signals for individual agents so that the agents collectively maximize some external utility measure. The COIN framework has also been applied to some real-world applications, see Wolpert, Tumer and Frank (1999) and Wolpert, Kirshner, Merz and Tumer (2000).

Learning methods based on MGs usually estimate the opponents' behavior in each state by applying the same RL algorithm that the opponent uses itself (this is possible if the actions of all agents as well as all rewards are fully observable). Keeping a copy of the opponent requires much space and learning the models of the opponents requires much computation. Therefore, it would be preferable to separate opponent modeling from the actual learning methods. In Uther and Veloso (2003), a simple fictitious play model is used to opponent modeling in MGs. Tesauro (2004), applies Q-learning for mixed strategies in game learning and uses Bayesian methods for opponent modeling.

4.6 Why to use MGs in multiagent reinforcement learning?

The principal idea behind MGs is to model the opponent and the agent's own, opponent dependent utility function and then act rationally based on this information. Compared to single-agent MDPs, storing additional parameters for opponent modeling and for the own utility function requires a considerably larger amount of space, and in the learning problem, learning these parameters requires more computation. So why should one use learning methods based on MGs instead of simpler single-agent MDPs?

In a multiagent environment, if the other agents' behavior converge, i.e. their action selection distributions become stationary in the limit, the normal Q-learning update rule will converge to the optimal Q-function with probability one. Additionally, if the agent uses a GLIE (Greedy in the Limit with Infinite Exploration) exploration policy and the best response policy is unique, it will converge in behavior with probability one (Littman, 2001b). However, two simultaneous single-agent Q-learners do not in general converge to mutual best responses. Even in simple MGs containing only one state, a best response policy can be stochastic.

In all cases, it is not adequate to use GLIE exploration policies. For example, if the state and action spaces are sufficiently small, it is possible to teach the whole utility function to the agents by sweeping multiple times through the state-actions space. In this case, a single-agent utility function represents expected values averaged over the opponent's actions. This can lead to suboptimal behavior in some cases. As an example, consider a simple two-agent example where the agents have the utility values presented in Game 4.1. Naturally, if multiagent reinforcement learning is used (there is a matrix of values for both agents), the agents learn exactly their own utility functions. If agent 1 is acting as a leader by selecting its action prior to agent 2, the equilibrium point is (a_2^1, a_2^2), leading to the payoffs $(2.0, 1.0)$. With single-agent learning, the agents learn the averaged payoff values that imply suboptimal behavior as they lead to payoff values $(0.0, 0.0)$.

	a_1^2	a_2^2	single(1)
a_1^1	$1.0, 2.0$	$0.0, 0.0$	0.5
a_2^1	$0.0, 0.0$	$2.0, 1.0$	1.0
single(2)	1.0	0.5	

Game 4.1: A simple stagegame for two agents. The first number in each cell is a payoff for agent 1 and the second number for agent 2. The row and column marked with the word single are averages over the opponent's action selections.

Each stagegame in an MG corresponds to a social interaction between agents. Therefore, in some cases, it is more desirable to learn these games and analyze them to acquire an overview of the problem. In addition, it is possible to classify these games automatically. Oppositely, single-agent learning methods could only provide average values and the real nature of the multiagent learning task would remain unclear.

Chapter 5

Example problems

This chapter briefly introduces three small example problems that are used for testing the presented multiagent reinforcement learning algorithms in the publications included in this thesis. Although the problems are quite simple, they are widely used for testing multiagent reinforcement learning algorithms and methods in the literature. The used example problems are the *grid world example*, the *simplified soccer game*, and *two pricing applications*.

5.1 Grid world example

Various grid world problems have been used for testing both single-agent and multiagent reinforcement learning methods in many works (see e.g. Sutton and Barto, 1998; Mitchell, 1997; Greenwald and Hall, 2003; Littman, 2001a). The problem used in this thesis, originally proposed by Hu and Wellman (1998), is a version of a robot navigation problem in which the agents learn to navigate through a grid world without colliding with each other or with other obstacles. The problem is particularly interesting due to its general-sum payoff structure, i.e. the interests of the agents are only partially conflicting.

In Publication I, two versions of the grid world problem are used for testing the proposed learning techniques. Version 1 contains only deterministic state transitions whereas version 2 has also two probabilistic transitions. The versions are similar to those in Hu and Wellman (1998), in which the problems were solved using a tabular symmetric multiagent reinforcement learning algorithm. In this thesis, both versions of the problem are solved by using the symmetric and asymmetric learning models. Moreover, in Publication II, the version 1 of the problem is solved also by using several numeric learning methods.

In both versions of the grid world problem, there is a grid world containing nine cells and two competing agents (Fig. 5.1). The agents start from the lower corners marked with 1 and 2, respectively, and on each round they can move to an adjacent cell (4-neighborhood). In version 1 of the problem, there are two distinct goal positions, one for each agent, and in version 2, the goal cell is the same for both agents. The agents get large positive payoffs when they reach the right goal positions. In the symmetric learning model both agents get small negative payoffs if they try to move to the same position, after which the agents are returned back to their original positions. In the asymmetric learning model, only agent 1 (leader) gets the negative payoff and is thereby trying to avoid the collision with its enforcements. Overall, the ultimate goal of the agents is to reach the goal cells using as few moves as possible.

In version 2 of the problem, there are two barriers above the initial cells 1 and 2. When an agent tries to move upwards from the start position, it gets through with probability 0.5, otherwise the barrier blocks the movement and the agent remains in the start position. This extension can be modeled as a stochastic state transition in the corresponding Markov game (MG).

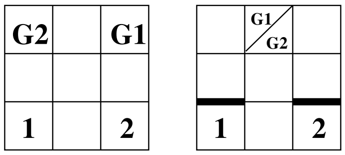


Figure 5.1: The game boards used in the two versions of the grid world problem. In both cases, the agents are initially located in the cells marked with the numbers **1** and **2**. The goal cells are marked with the symbols **G1** and **G2**. In version 2 of the problem, there are barriers above the start cells (marked with thick lines).

The problem can be characterized with the following MG:

- A state in this problem is a pair $s = (p_1, p_2)$, consisting of the positions of the agents. Hence, the state space of this example consists of 9 x 9=81 states.
- The agents get positive payoffs of 0.9 when they find the right goal cells.
- The action set for both agents is $A^i = \{\text{Left}, \text{Right}, \text{Up}, \text{Down}\}, i = 1, 2$. The agents are restricted to stay on the game board.
- The discount factor γ is 0.99.
- In the asymmetric model, agent 1 is acting as the leader.
- In the symmetric model, if the agents collide, both agents get negative rewards of -0.1. Only the leader gets this negative reward in the asymmetric model.

When one of the agents reaches the goal position, both agents are moved back to their initial positions and the learning round is continued as a new episode. The learning was repeated 50 times in every test case. Some of the equilibrium paths generated by the symmetric and the asymmetric learning models in version 1 are shown in Fig. 5.2. In Fig. 5.3, the two equilibrium paths in version 2 are shown.

The averaged payoff values from the test runs are shown in Table 5.1. In version 1, both learning models performed equally and found optimal paths. In version 2, the symmetric model found both of the optimal paths illustrated in Fig. 5.3, whereas the asymmetric model

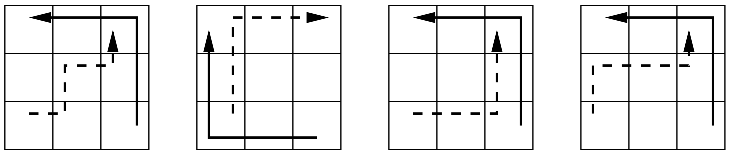


Figure 5.2: Some equilibrium paths in version 1 of the grid world problem.

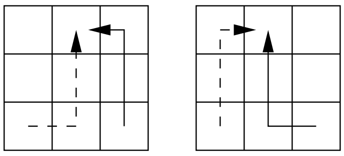


Figure 5.3: Equilibrium paths in version 2 of the grid world problem.

found only the path on the right. The reason for this phenomenon is that only the leader gets negative feedback from a collision, which motivates the leader to move upwards in the start state.

Table 5.1: The averaged payoff values from 50 runs in the form (agent 1, agent 2).

	version 1	version 2
symmetric	0.87, 0.87	0.51, 0.79
asymmetric	0.87, 0.87	0.37, 0.88

In Publication III, the state space of the grid world problem is divided into two subspaces: one containing states with matrix games (complex states) and the other containing states with the utility values for the learning agent only (simple states). The main idea is that the complex states model the "critical" parts of the state space. In this problem, the complex states are states where the distance between the agents is at most two cells. The simple states then model the "easy" parts of the state space where there is no interaction between the agents. In the grid world problems, if the distance between the agents is exactly two cells, there exist joint actions that lead to a collision between the agents and therefore it is justified to apply multiagent reinforcement learning in these complex states. The grid world problem used to test this hybrid model is similar to the previously introduced version 1 of the problem except that the size of the grid is 4 x 4 cells.

The hybrid model was tested with both symmetric and asymmetric learning models in Publication III. The actual learning was carried out using a Q-learning type off-policy learning method (details are given in Publication III). In each case, the learning was repeated 50 times and each learning round took 100000 episodes. The averaged learning times (CPU time in seconds) are presented in Table 5.2. From the table it can be seen that the learning procedure is significantly faster with the hybrid learning model. In addition, the learning was much faster with the asymmetric than with the symmetric learning model.

Table 5.2: Averaged learning times (CPU time in seconds) from 50 learning rounds.

	normal	hybrid
symmetric	237	180
asymmetric	78	48

5.2 Simplified soccer game

The aim of Publication IV is to provide a direct policy gradient method for team MGs, i.e. MGs where all the agents share the same utility function. The proposed method is tested with a simple soccer game that was originally proposed by Peshkin, Kim, Meuleau and Kaelbling (2000). In this game, there are three players (agents): one fixed-strategy opponent and two learning agents that constitute a team. One of the three agents holds the ball at any time. The game is played on a 5×6 field illustrated in Fig. 5.4. In this figure, the cell marked with **G1** is the goal for the learning agents and the cell **G2** for the fixed-strategy agent. The agents are capable of moving to the four cardinal directions or staying in the current cell. Only one agent can be in a certain cell simultaneously. The agent having the ball loses it when colliding with another agent. In addition, the learning agents are capable of passing, within two cells, the ball to their team mate.

A state consists of the agents' positions and the ball possession information. Initially the fixed strategy agent is located in the left half of the field and the learning agents in the right half of the field. The ball possession is selected randomly. The game ends when the agent possessing the ball reaches a goal cell. The cell **G1** produces the payoff of 1 and **G2** the payoff of -1. The players are then returned back to random initial positions. After the agents select their actions, using the Stackelberg equilibrium concept or by sampling from the joint policy function, the actions are carried out in random order. This induces stochastic state transitions to the MG. The fixed strategy player always moves toward the agent possessing the ball until it gets the ball, after which it starts to move directly toward the goal cell **G2**.

The model was taught with 50000 games. The discount factor was $\gamma = 0.9$ and the maximum length of a game was restricted to 50 moves. Additionally, the model was tested with 1000 games. In Fig. 5.5, the average number of wins (averaged over 20 test runs) is plotted against the number of training games. From this figure, it can be seen that the number of wins increases along the number of games. However, the system learns very fast in the beginning of the learning process, and later the learning continues but it is not as dramatic. When the system has been trained with more than 40000 games, there is a small drop in the number of wins. This is due to the randomness in the system, namely stochastic state transitions and random initial configurations.

5.3 Pricing scenarios

Two pricing scenarios, the flat pricing problem and the two-layer pricing problem, are used to test proposed methods in three of the publications included in the thesis. In Publication II, different numeric solution techniques are applied to the simple (flat) pricing scenario

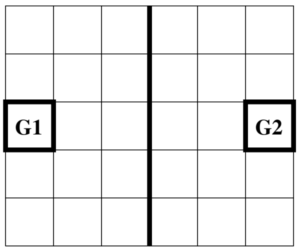


Figure 5.4: The game field in the soccer example. The cell **G1** is the goal for the learning agents and the cell **G2** for the fixed strategy opponent.

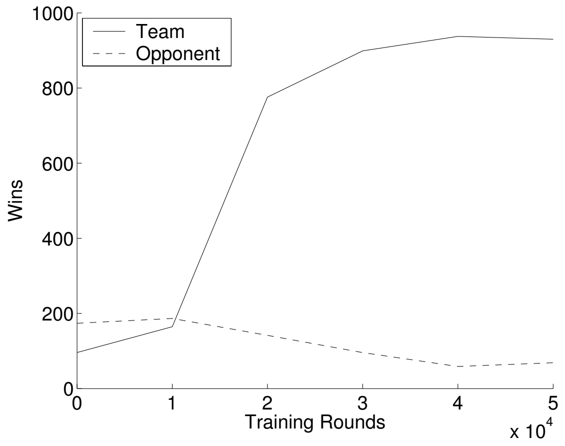


Figure 5.5: The number of wins plotted against the number of the training rounds.

and in Publication I, a two-layer pricing problem is solved by using the asymmetric multiagent reinforcement learning method. Moreover, both pricing scenarios are discussed more thoroughly in Publication V. Overall, due to different cost structures commonly arising in economical systems, asymmetric equilibrium concepts are extensively used to analyze these systems in operation research literature (see e.g. Ho, Luh and Muralidharan, 1981; Salman and Cruz, 1981). In this thesis, the problem is much harder; the goal of the agents is to learn to behave optimally towards an unknown opponent.

In both scenarios, there are two competing agents (brokers) that sell identical products and compete against each other on the basis of price. At each time step, one of the brokers decides its new price based on the opponent's, i.e. the other broker's, current price. After the prices have been set, the customer either buys a product from the broker with the cheapest price or decides not to buy the product at all. The objective of the agents is to maximize their profits. The interaction between the two brokers is modeled as an asymmetric multiagent reinforcement learning model. Additionally, the flat pricing scenario is extended to the hierarchical pricing problem of three agents, in which one of the agents is acting as a supplier that sells products to the brokers.

5.3.1 Flat pricing problem

Tesauro and Kephart (1999) modeled the interaction between two brokers as a single-agent reinforcement learning problem in which the goal of the learning agent is to find the pricing strategy that maximizes its long time profits. Additionally, reinforcement learning aids the agents to prevent "price wars", i.e. repeated price reductions among the brokers. As a consequence of a price war, the prices would go very low and the overall profits would be small. Tesauro and Kephart reported very good performance of the approach when one of the brokers keeps its pricing strategy fixed. However, if both brokers try to learn simultaneously, the Markov property assumed in the theory of Markov decision processes does not hold and the learning system may encounter serious convergence problems. However, the learning agents achieved a good performance with some pricing scenarios and the methods were even extended to apply function approximators such as neural networks and regression trees in (Tesauro, 2001; Sridharan and Tesauro, 2000). A minimax-style algorithm was applied to the pricing problem in (Tesauro and Kephart, 1998).

In this thesis, the pricing scenario is modeled as an asymmetric MG. In Publication V, the system is tested with two economical models: the *Shopbot model* (Greenwald and Kephart, 1999) and *Price-Quality model* (Sairamesh and Kephart, 1998). In this chapter, for brevity, only the Shopbot model is discussed.

In the Shopbot model, the customer buys the product from the broker having the lowest price. At each time step, after the customer has done his purchase decision, both brokers get their immediate profits according to the utility functions defined as follows:

$$u^1(p^1, p^2) = \begin{cases} p^1 - c & \text{if } p^1 \le p^2 \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

and

$$u^2(p^1, p^2) = \begin{cases} p^2 - c & \text{if } p^1 > p^2 \\ 0 & \text{otherwise,} \end{cases} \tag{5.2}$$

where $p^1, p^2 \in P$ are the current prices of broker 1 and broker 2, respectively, and $c \in [0, 1]$ is the fixed marginal cost of the product. In this thesis, all prices lie in the unit interval and the parameter $c = 0.2$.

The basic assumption is that the brokers do not make their decisions simultaneously, i.e. there is an ordering among the decision makers. Hence, the system is modeled with the following asymmetric MG:

- The state is the current price of broker 2.
- Broker 1 is acting as the leader and decides its price prior to broker 2. Hence, as the state is the current price of broker 2, the utility of broker 1 depends only on its price selection and the current state.
- Broker 2 is the follower and its utility value depends on the leader's enforcement and its own price selection.

At each time step, broker 1 calculates the Stackelberg equilibrium point of the matrix game associated with the current state and makes its pricing decision based on this solution. After that, broker 1 announces its price decision to broker 2 who, in its turn, maximizes its utility value based on this enforcement. This process is illustrated in Fig. 5.6.

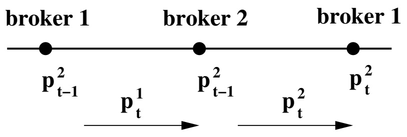


Figure 5.6: Timeline of the price decisions in the flat pricing problem. The price symbols below the dots describe the states and symbols above the arrows represent the price decisions.

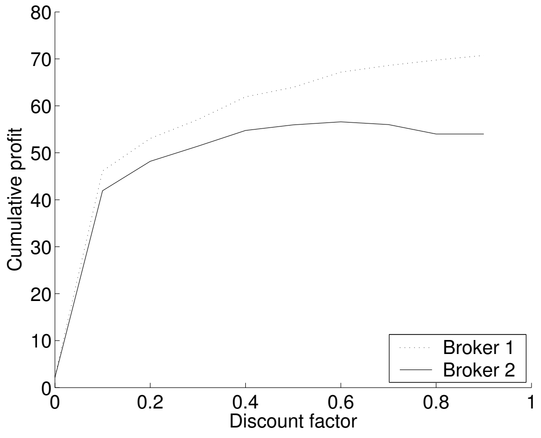


Figure 5.7: The averaged profits in the flat pricing model with the Shopbot pricing function. All data points are averages of 1000 test runs, each containing 100 pricing decisions for both agents.

In the test runs, the number of different pricing options was set to 25 for both agents. During training, each state-actions tuple was visited 1000 times. In the testing phase, the initial state (price of broker 2) was selected randomly and one test run consisted of 100 pricing decisions per broker. In Fig. 5.7, the cumulative profit (average from 1000 test runs) of each agent is plotted against the discount factor γ in the case of the Shopbot pricing model. As can be seen from the figure, the average profit of broker 1 grows monotonically as the discount factor increases. Also the profit of broker 2 increases albeit not monotonically. Moreover, even the use of a small discount factor $\gamma = 0.1$, corresponding to a very shallow lookahead, leads to relatively high profits compared to $\gamma = 0.0$. The use of higher discount factors increases profits further but the growth is not so dramatic.

5.3.2 Two-layer pricing problem

In addition to the flat pricing problem, the two-layer pricing problem contains a supplier that sells products to both of the brokers. At each time step, one of the brokers decides its new price based on the opponent's (other broker) current price and the price set by the supplier. The supplier, in its turn, decides its action based on the asymmetric solution concept. After the prices have been set, the customer either purchases a product from a broker or decides not to buy the product at all. After the customer's decision, the brokers get their profits according to their immediate reward functions presented in Eqs. (5.3) and (5.4). The utility values for the supplier are shown in Eqs. (5.5) and (5.6) when the brokers 1 and 2 are charged, respectively.

$$u^1(p^1, p^2, s; l) = \begin{cases} p^1 - s & \text{if } p^1 \leq p^2 \text{ and } s < lp^1 \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

$$u^2(p^1, p^2, s; l) = \begin{cases} p^2 - s & \text{if } p^1 > p^2 \text{ and } s < lp^2 \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

$$u^{s1}(p^1, p^2, s; l) = \begin{cases} s - c & \text{if } p^1 \leq p^2 \text{ and } s < lp^1 \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

$$u^{s2}(p^1, p^2, s; l) = \begin{cases} s - c & \text{if } p^1 > p^2 \text{ and } s < lp^2 \\ 0 & \text{otherwise.} \end{cases} \tag{5.6}$$

In Eqs. (5.3)–(5.6), p^1 and p^2 are the prices of the brokers 1 and 2, respectively, s is the price of the supplier and $l \in [0, 1]$ is the largest fraction of the broker's price that the broker is willing to pay to the supplier. As in the flat pricing problem, c is a fixed marginal cost of the product. c could also be associated with some quality parameter, perhaps different for each broker. However, in this thesis, the parameter has the same fixed value for each broker.

At each time step, the customer compares the prices and purchases the product from the broker only if its price decision is lower than the current price of its competitor. In addition, if the supplier is charging too much from the broker (expected profit of the broker is too low), the broker does not buy the product from the supplier and the utility drops to zero for both the supplier and the broker.

In this problem, reinforcement learning is used to aid the agents in anticipating the long-time consequences of their price decisions on both levels of the agent hierarchy. The supplier does not know the actual value of l used in Eqs. (5.3)–(5.6) and therefore it is justified also to use learning, e.g. reinforcement learning, also for the supplier.

The learning task is simplified by assuming that broker 2 keeps its pricing strategy fixed, i.e. it decides its price based on the immediate utility value defined in Eq. (5.4). Furthermore, the supplier also keeps its pricing strategy fixed with broker 2. Fig. 5.8 illustrates this relationship.

In the corresponding MG, the state is the opponent's (other broker's) last price and the action is the current price decision. As broker 2 uses a fixed strategy, there is no need for a game between the brokers. The parameter l has a value of 0.8 and the producing cost for the supplier is $c = 0.2$ per product. In addition, the maximum price allowed to the supplier is 0.8. The training phase was conducted similarly as with the flat pricing model.

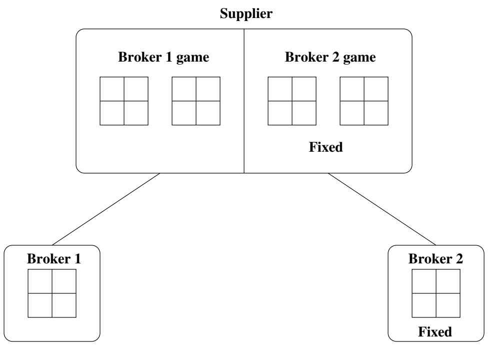


Figure 5.8: Supplier-broker relationship.

In the testing phase, the initial prices were selected randomly and one test run consisted of 100 pricing decisions per broker. In Fig. 5.9, the cumulative profit (average from 1000 test runs) of each agent is plotted against the discount factor γ. As can be seen from this figure, the average profit of the supplier grows monotonically as the discount factor increases. The brokers also learn a pricing strategy that leads to a notable growth in the profits compared to the myopic case ($\gamma = 0$). The optimizing broker (broker 1) rises its price to the maximum in some situations and therefore has slightly lower profits than the static broker. However, the cumulative profits are much higher also for broker 1 when compared to the myopic case.

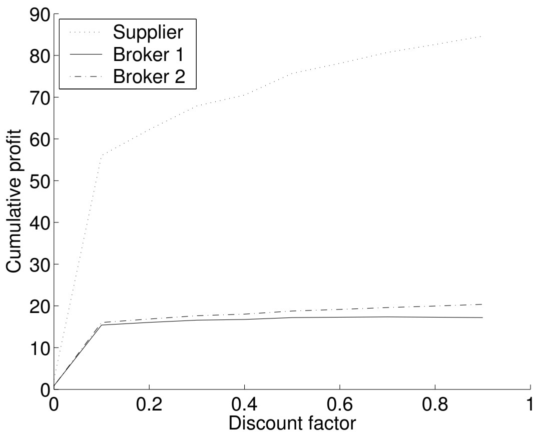


Figure 5.9: Averaged profits in the two-layer pricing model. All data points are averages of 1000 test runs each containing 100 pricing decisions for both agents. The maximal possible profit is 100 for the brokers and 200 for the supplier.

Chapter 6

Conclusions

Most real-life situations include social interactions between multiple natural or artificial actors. It is often crucial to be able to anticipate the behavior of the other actors in order to satisfy one's own goals. Recreational games, e.g. board games, are good examples of these interactions between humans. A mathematical tool for dealing with these interactions is game theory.

In many problems, the nature of the interaction is not known a priori; the exact structure of the game should be learned from the history of interactions between agents. Therefore, the fusion of traditional game theory and various machine learning techniques is a very active research area at the present moment. Although multiagent machine learning techniques have been studied for a long time, the problem has manifested to be very hard. As a result, research based on multiagent reinforcement learning is still in the "basic research" phase and most of the real-world multiagent applications use methods designed originally for single-agent domains. In this thesis, various aspects of multiagent reinforcement learning are studied and several extensions to single-agent reinforcement learning for multiagent domains are proposed.

One of the main contributions of this thesis is the application of the asymmetric Stackelberg equilibrium concept for evaluating stagegame values in Markov games (MGs). This procedure leads to deterministic optimal policies and is much faster than the use of the symmetric equilibrium concept. Moreover, it is possible to use a combination of the asymmetric and symmetric equilibrium concepts in the learning process and thereby to cover a wide variety of different problem settings.

Applying reinforcement learning methods for solving complex problems requires the use of function approximators. In this thesis, two approaches to numeric approximation in multiagent reinforcement learning tasks are studied. In the first approach, the Q-function is approximated with a function approximator, which is then used for estimation of the optimal policies. In the second approach, the policy function is parameterized directly and the optimal parameter values are then sought by using a gradient based method.

In many real-world applications where reinforcement learning techniques are deployed to multiagent systems, single-agent reinforcement learning is directly applied. Although some principal assumptions behind these learning models are violated, the methods work surprisingly well in many cases. In this thesis, a hybrid model is proposed in which the interaction between the agents is modeled only in some predefined states and the normal single-agent learning model in the remaining states. This can considerably reduce the number of free parameters in the model and thus speed up learning, while maintaining the expression power of MGs in the critical states.

The nature of the problem in question stipulates the equilibrium concept in MGs. In this thesis, the learning methods are tested with several example problems. In some of these tests, the roles of the learning agents are symmetric, leading to symmetric equilibrium concepts, whereas in other tests, the agents have distinct roles leading to the asymmetric equilibrium concept. The pricing problems discussed in the thesis are examples of problems, where the roles of the agents are naturally different due to the cost structure. Additionally, both the symmetric and asymmetric learning methods are applied to simple grid world problems and their convergence speeds are compared. The asymmetric equilibrium concept provides one justification for the use of the MaxMax operator in multiagent Q-learning with team games. Based on this operator, a policy gradient method is extended to multiagent domains and tested with a simplified soccer game.

Since MGs generally have a huge number of free parameters to learn and there are several possible solution concepts with different features, MGs have been mainly of theoretical interest to researchers in the field of reinforcement learning. However, there exist equilibrium concepts, for example the Stackelberg equilibrium concept, that make it possible to evaluate games very quickly and therefore to accelerate the learning procedure considerably. On the other hand, this approach requires that the learning agents accept their roles and with general-sum games, it is not possible to provide general convergence proofs. Therefore it remains an open question to find a learning method that is guaranteed to converge to an equilibrium policy with general-sum problems. When the stagegames in an MG have been learned, it is possible to classify and study the properties of these games and acquire relevant information about the underlying process. This classification could be accomplished either manually or automatically by applying the methods for classification and clustering. The interesting future research direction is to study the encoding of the games, i.e. how the stagegames can be provided to the classification methods.

Bibliography

Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation, *Proceedings of the Twelft International Conference on Machine learning (ICML-1995)*, Tahoe City, CA, pp. 30–37.

Baird, L. C. and Moore, A. (1999). Gradient descent for general reinforcement learning, *Advances in Neural Information Processing Systems (NIPS-1998)*, Denver, CO, pp. 968–974.

Banerjee, B., Sen, S. and Peng, J. (2001). Fast concurrent reinforcement learners, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001)*, Seattle, WA, pp. 825–832.

Bard, J. F. (1998). *Practical Bilevel Optimization—Algorithms and Applications*, Kluwer Academic Publishers.

Barto, A. G., Bradtke, S. J. and Singh, S. P. (1995). Learning to act using real-time dynamic programming, *Artificial Intelligence* **72**(1–2): 81–138.

Basar, T. and Olsder, G. J. (1982). *Dynamic Noncooperative Game Theory*, Vol. 160 of *Mathematics in Science and Engineering*, Academic Press Inc. (London) Ltd.

Bellman, R. E. (1957a). *Dynamic Programming*, Princeton University Press.

Bellman, R. E. (1957b). A Markov decision process, *Journal of Mathematical Mechanics* **6**: 679–684.

Bellman, R. E. (1961). *Adaptive Control Processes: a Guided Tour*, Princeton University Press.

Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*, Athena Scientific.

Borel, E. (1953a). On games that involve chance and the skill of the players, *Econometrica* **21**(1): 101–115.

Borel, E. (1953b). On systems of linear forms of skew symmetric determinant and the general theory of play, *Econometrica* **21**(1): 116–117.

Borel, E. (1953c). The theory of play and integral equations with skew symmetric kernels, *Econometrica* **21**(1): 97–100.

Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes, *Proceedings of the Sixth Conference on Theoretical Aspects of Rationality and Knowledge (TARK-1996)*, De Zeeuwse Stromen, The Netherlands, pp. 195–210.

Bower, G. H. and Hilgard, E. R. (1981). *Theories of Learning*, Prentice-Hall.

Bowling, M. (2000). Convergence problems of general-sum multiagent reinforcement learning, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, Stanford, CA, pp. 89–94.

Bowling, M. and Veloso, M. M. (2002a). Multiagent learning using a variable learning rate, *Artificial Intelligence* **136**(2): 454–460.

Bowling, M. and Veloso, M. M. (2002b). Scalable learning in stochastic games, *Proceedings of the AAAI-02 Workshop on Game Theoretic and Decision Theoretic Agents*, Edmonton, Canada, pp. 11–18.

Boyan, J. A. (1992). *Modular neural networks for learning context-dependent game strategies*, Master's thesis, Cambridge University.

Carmel, D. and Markovitch, S. (1996). Incorporating opponent models into adversary search, *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, Portland, OR, pp. 120–125.

Carmel, D. and Markovitch, S. (1998). Model-based learning of interaction strategies in multi-agent systems, *Journal of Experimental and Theoretical Artificial Intelligence* **10**(3): 309–332.

Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems, *Proceedings of the Fifteenth National Conference of Artificial Intelligence (AAAI-98)*, Madison, WI, pp. 746–752.

Conitzer, V. and Sandholm, T. W. (2003a). AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents, *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, Washington, DC, pp. 83–90.

Conitzer, V. and Sandholm, T. W. (2003b). BL-WoLF: A framework for loss-bounded learnability in zero-sum games, *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, Washington, DC, pp. 91–98.

Conitzer, V. and Sandholm, T. W. (2003c). Complexity results about Nash equilibria, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-2003)*, Acapulco, Mexico, pp. 765–771.

Cottle, R. W., Stone, R. E. and Pang, J.-S. (1992). *The Linear Complementarity Problem*, Academic Press.

Cournot, A. A. (1897). *Researches into the Mathematical Principles of the Theory of Wealth*, Macmillan.

Crites, R. H. (1996). *Large-Scale Dynamic Optimization Using Teams of Reinforcement Learning Agents*, PhD thesis, University of Massachusetts Amherst.

Cziko, G. (1995). *Without miracles : universal selection theory and the second Darwinian revolution*, MIT Press.

Dantzig, G. B. (1963). *Linear Programming and Extensions*, Princeton University Press.

Dean, T., Kaelbling, L. P., Kirman, J. and Nicholson, A. (1993). Planning with deadlines in stochastic domains, *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-93)*, Washington, DC, pp. 574–579.

Dennett, D. C. (1978). Why the law-of-effect will not go away, *Brainstorms*, Bradford/MIT Press, pp. 71–89.

Edgeworth, F. Y. (1881). *Mathematical Psychics: An Essay on the Application of Mathematics to the Moral Sciences*, Kegan Paul.

Filar, J. A. and Vrieze, K. (1997). *Competitive Markov Decision Processes*, Springer-Verlag.

Fudenberg, D. and Levine, D. K. (1998). *The Theory of Learning in Games*, MIT Press.

Gmytrasiewicz, P. J. and Durfee, E. H. (1995). A rigorous, operational formalization of recursive modeling, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-1995)*, Menlo Park, CA, pp. 125–132.

Gmytrasiewicz, P. J. and Durfee, E. H. (2000). Rational coordination in multi-agent environments, *Autonomous Agents and Multi-Agent Systems* **3**(4): 319–350.

Greenwald, A. and Hall, K. (2003). Correlated-Q learning, *Proceedings of the Eighteenth International Conference on Machine Learning (ICML-2003)*, Washington, DC, pp. 242–249.

Greenwald, A. R. and Kephart, J. O. (1999). Shopbots and pricebots, *Proceedings of the IJCAI-1999 Workshop on Agent Mediated Electronic Commerce*, Stockholm, Sweden, pp. 1–23.

Ho, Y.-C., Luh, P. B. and Muralidharan, R. (1981). Information structure, Stackelberg games, and incentive controllability, *IEEE Transactions on Automatic Control* **AC-26**(2): 454–460.

Hu, J. (1999). *Learning in Dynamic Noncooperative Multiagent Systems*, PhD thesis, The University of Michigan.

Hu, J. and Wellman, M. P. (1998). Multiagent reinforcement learning: Theoretical framework and an algorithm, *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-1998)*, Madison, WI, pp. 242–250.

Hu, J. and Wellman, M. P. (2000). Experimental results of multiagent reinforcement learning, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, Stanford, CA, pp. 407–414.

Hu, J. and Wellman, M. P. (2002). Learning about other agents in a dynamic multiagent system, *Cognitive Systems Research* **2**(1): 67–79.

Hu, J. and Wellman, M. P. (2003). Nash Q-learning for general-sum stochastic games, *Journal of Machine Learning Research* **4**: 1039–1069.

Huang, P. and Sycara, K. (2003). Multi-agent learning in extensive games with complete information, *Proceedings of the Second International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS-2003),*, Melbourne, Australia, pp. 701–708.

IMS (2004). IMSL Numeric Library. http://www.vni.com/.

Kaelbling, L. P., Littman, M. L. and Moore, A. W. (1996). Reinforcement learning: A survey, *Journal of Artificial Intelligence Research* **4**: 237–285.

Kapetanakis, S. and Kudenko, D. (2002a). Improving on the reinforcement learning of coordination in cooperative multi-agent systems, *Proceedings of the AISB-2002 Symposium on Adaptive Agents and Multi-Agent Systems*, London, UK.

Kapetanakis, S. and Kudenko, D. (2002b). Reinforcement learning of coordination in cooperative multi-agent systems, *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-02)*, Edmonton, Alberta, Canada, pp. 326–331.

Kapetanakis, S., Kudenko, D. and Strens, M. (2003). Learning to coordinate using commitment sequences in cooperative multiagent-systems, *Proceedings of the AISB-2003 Symposium on Adaptive Agents and Multi-Agent Systems*, Aberystwyth, UK.

Konda, V. R. and Tsitsiklis, J. N. (2000). Actor-critic algorithms, *Advances in Neural Information Processing Systems (NIPS-1999)*, Denver, CO, pp. 1008–1014.

Könönen, V. J. (2003a). Asymmetric multiagent reinforcement learning, *Proceedings of the 2003 WIC International Conference on Intelligent Agent Technology (IAT-2003)*, Halifax, Canada, pp. 336–342.

Könönen, V. J. (2003b). Gradient based method for symmetric and asymmetric multiagent reinforcement learning, *Proceedings of the Fourth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL-2003)*, Hong Kong, China, pp. 68–75.

Könönen, V. J. (2003c). Policy gradient method for multiagent reinforcement learning, *Proceedings of the second International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS-2003)*, Singapore. CD-ROM.

Kuhn, H. W. (1953). Extensive games and the problem of information, *in* H. W. Kuhn and A. W. Tucker (eds), *Contributions to the Theory of Games*, Vol. 2, Princeton University Press, pp. 193–216.

Lauer, M. and Riedmiller, M. (2000). An algorithm for distributed reinforcement learning in cooperative multi-agent systems, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, Stanford, CA, pp. 535–542.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning, *Proceedings of the Eleventh International Conference on Machine Learning (ICML-1994)*, New Brunswick, NJ, pp. 157–163.

Littman, M. L. (1996). *Algorithms for Sequential Decision Making*, PhD thesis, Brown University.

Littman, M. L. (2001a). Friend-or-Foe Q-learning in general-sum games, *Proceedings of the Eighteenth International Conference on Machine Learning (ICML-2001)*, Williamstown, MA, pp. 322–328.

Littman, M. L. (2001b). Value-function reinforcement learning in Markov games, *Cognitive Systems Research* **2**(1): 55–66.

Littman, M. L. and Stone, P. (2001). Implicit negotiation in repeated games, *Proceedings of The Eighth International Workshop on Agent Theories, Architectures, and Languages (ATAL-2001)*, Seattle, WA, pp. 393–404.

Luce, R. D. and Raiffa, H. (1957). *Games and Decisions: Introduction and Critical Survey*, John Wiley and Sons.

McKelvey, R. D. and McLennan, A. (1996). Computation of equilibria in finite games, *in* H. M. Amman, D. A. Kendrick and J. Rust (eds), *Handbook of Computational Economics*, Vol. 1, Elsevier Science, pp. 87–142.

Mitchell, T. M. (1997). *Machine Learning*, McGraw-Hill.

Moore, A. W. and Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time, *Machine Learning* **13**(1): 103–130.

Murty, K. G. (1978). Computational complexity of complementary pivot methods, *Mathematical Programming Study* **7**: 61–73.

Myerson, R. B. (1991). *Game Theory: Analysis of Conflict*, Harvard University Press.

NAG (2004). Numerical Algorithms Group (NAG). http://www.nag.co.uk.

Nash, Jr., J. F. (1950a). The bargaining problem, *Econometrica* **18**(2): 155–162.

Nash, Jr., J. F. (1950b). Equilibrium points in N-person games, *Proceedings of National Academy of Sciences of the United States of America* **36**: 48–49.

Nash, Jr., J. F. (1953). Two person cooperative games, *Econometrica* **21**(1): 128–140.

Pavlov, I. P. (1984). *Conditioned Reflexes*, Dover Publications.

Peshkin, L., Kim, K.-E., Meuleau, N. and Kaelbling, L. P. (2000). Learning to cooperate via policy-search, *Proceedings of the Sixteenth Conference on Uncertainty in Artifical Intelligence (UAI-2000)*, Stanford, CA, pp. 489–496.

Robbins, H. and Monro, S. (1951). A stochastic approximation method, *Annals of Mathematical Statistics* **22**(3): 400–407.

Rummery, G. A. and Niranjan, M. (1994). On-line Q-learning using connectionist systems, *Technical Report CUED/F-INFENG/TR166*, Cambridge University, Engineering Department.

Sairamesh, J. and Kephart, J. O. (1998). Price dynamics of vertically differentiated information markets, *Proceedings of the First International Conference on Information and Computational Economics (ICE'98)*, Charleston, SC, pp. 28–36.

Salman, M. A. and Cruz, Jr., J. B. (1981). An incentive model of duopoly with government coordination, *Automatica* **17**(6): 821–829.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers, *IBM Journal of Research and Development* **3**(3): 210–229.

Sandholm, T. W. and Crites, R. H. (1995). On multiagent Q-learning in a semi-competitive domain, *Proceedings of the IJCAI-1995 Workshop on Adaption and Learning in Multi-Agent Systems*, Montréal, Canada, pp. 191–205.

Schraudolph, N. N., Dayan, P. and Sejnowski, T. J. (1993). Temporal difference learning of position evaluation in the game of go, *Advances in Neural Information Processing Systems (NIPS-1993)*, Denver, CO, pp. 817–824.

Selten, R. (1975). Reexamination of the perfectness concept for equilibrium points in extensive games, *International Journal of Game Theory* **4**: 25–55.

Shapley, L. S. (1953). Stochastic games, *Proceedings of National Academy of Sciences of the United States of America* **39**: 1095–1100.

Sheppard, J. W. (1997). *Multi-Agent Reinforcement Learning in Markov Games*, PhD thesis, The Johns Hopkins University.

Singh, S. P., Jaakkola, T. S., Littman, M. L. and Szepesvári, C. (2000). Convergence results for single-step on-policy reinforcement-learning algorithms, *Machine Learning* **38**(3): 287–308.

Singh, S. P., Kearns, M. and Mansour, Y. (2000). Nash convergence of gradient dynamics in general-sum games, *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, Stanford, CA, pp. 541–548.

Sridharan, M. and Tesauro, G. (2000). Multi-agent Q-learning and regression trees for automated pricing decisions, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, Stanford, CA, pp. 927–934.

Suematsu, N. and Hayashi, A. (2002). A multiagent reinforcement learning algorithm using extended optimal response, *Proceedings of the First International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS-2002)*, Bologna, Italy, pp. 370–377.

Sun, R. and Qi, D. (2000). Rationality assumptions and optimality of co-learning, *Proceedings of the Third Pacific Rim International Workshop on Multi-Agents (PRIMA-2000)*, Melbourne, Australia, pp. 61–75.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences, *Machine Learning* **3**(1): 9–44.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming, *Proceedings of the Seventh International Conference on Machine Learning (ICML-1990)*, Austin, TX, pp. 216–224.

Sutton, R. S. (1991). Planning by incremental dynamic programming, *Proceedings of the Eighth International Workshop on Machine Learning (ML-1991)*, Evanston, IL, pp. 353–357.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*, MIT Press.

Sutton, R. S., Barto, A. G. and Anderson, C. W. (1983). Neuronlike elements that can solve difficult learning control problems, *IEEE Transactions on Systems, Man, and Cybernetics* **13**: 835–846.

Sutton, R. S., McAllester, D., Singh, S. P. and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation, *Advances in Neural Information Processing Systems (NIPS-1999)*, Denver, CO, pp. 1057–1063.

Szepesvári, C. and Littman, M. L. (1999). A unified analysis of value-function-based reinforcement-learning algorithms, *Neural Computation* **11**(8): 2017–2060.

Tesauro, G. (1992). Practical issues in temporal difference learning, *Machine Learning* **8**(3–4): 257–277.

Tesauro, G. (2001). Pricing in agent economies using neural networks and multi-agent Q-learning, *in* R. Sun and C. Giles (eds), *Sequence Learning: Paradigms, Algorithms, and Applications*, Vol. 1828 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 288–307.

Tesauro, G. (2004). Extending Q-learning to general adaptive multi-agent systems, *Advances in Neural Information Processing Systems (NIPS-2003)*, Vancouver, British Columbia, Canada, pp. 871–878.

Tesauro, G. and Kephart, J. O. (1998). Foresight-based pricing algorithms in an economy of software agents, *Proceedings of the First International Conference on Information and Computational Economics (ICE'98)*, Charleston, SC, pp. 37–44.

Tesauro, G. and Kephart, J. O. (1999). Pricing in agent economies using multi-agent Q-learning, *Proceedings of the Workshop on Game Theoretic and Decision Theoretic Agents (GTDT'99)*, London, UK, pp. 71–86.

Thorndike, E. L. (1911). *Animal Intelligence*, Macmillan.

Thrun, S. (1994). Learning to play the game of chess, *Advances in Neural Information Processing Systems (NIPS-1994)*, Denver, CO, pp. 1069–1076.

Uther, W. and Veloso, M. M. (2003). Adversarial reinforcement learning, *Technical Report CMU-CS-03-107*, Carnegie Mellon University.

Verbeeck, K., Nowé, A., Lenaerts, T. and Parent, J. (2002). Learning to reach the Pareto optimal Nash equilibrium as a team, *Proceedings of the Fifteenth Australian Joint Conference on Artificial Intelligence (AI-2002)*, Canberra, Australia, pp. 407–418.

von Neumann, J. (1959). On the theory of games of strategy, *in* A. W. Tucker and R. Luce (eds), *Contributions to the Theory of Games*, Vol. 4, Princeton University Press, pp. 13–42.

von Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behavior*, Princeton University Press.

von Stackelberg, H. (1934). *Marktform und Gleichgewicht (in German)*, Springer-Verlag.

von Stackelberg, H. (1952). *The Theory of Market Economy*, Oxford University Press.

von Stengel, B. (2002). Computing equilibria for two-person games, *in* R. J. Aumann and S. Hart (eds), *Handbook of Game Theory*, Vol. 3, North-Holland.

Wang, X. and Sandholm, T. W. (2003). Reinforcement learning to play an optimal Nash equilibrium in team Markov games, *Advances in Neural Information Processing Systems (NIPS-2002)*, Vancouver, British Columbia, Canada, pp. 1603–1610.

Wang, X. and Sandholm, T. W. (2004). Learning near-Pareto-optimal conventions in polynomial time, *Advances in Neural Information Processing Systems (NIPS-2003)*, Vancouver, British Columbia, Canada, pp. 863–870.

Watkins, C. J. (1989). *Learning from Delayed Rewards*, PhD thesis, Cambridge University.

Watkins, C. J. and Dayan, P. (1992). Q-learning, *Machine Learning* **8**(3–4): 279–292.

Weaver, L. and Tao, N. (2001). The optimal reward baseline for gradient-based reinforcement learning, *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI-2001)*, Seattle, WA, pp. 538–545.

Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits, *IRE Western Electric Show and Convention Record* **4**: 96–104.

Williams, R. J. (1988). Toward a theory of reinforcement-learning connectionist systems, *Technical Report NU-CCS-88-3*, College of Computer Science, Northeastern University.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning, *Machine Learning* **8**(3–4).

Witten, I. H. (1977). An adaptive optimal controller for discrete-time Markov environments, *Information and Control* **34**(4): 286–295.

Wolpert, D. H., Kirshner, S., Merz, C. J. and Tumer, K. (2000). Adaptivity in agent-based routing for data networks, *Proceedings of the Fourth International Conference on Autonomous Agents (Agents 2000)*, Barcelona, Catalonia, Spain, pp. 396–403.

Wolpert, D. H. and Tumer, K. (2000). An introduction to collective intelligence, *Technical Report NASA-ARC-IC-99-63*, NASA Ames Research Center.

Wolpert, D. H., Tumer, K. and Frank, J. (1999). Using collective intelligence to route Internet traffic, *Advances in Neural Information Processing Systems (NIPS-1998)*, Denver, CO, pp. 952–958.

Zachrisson, L. E. (1964). Markov games, *in* M. Dresher, L. S. Shapley and A. W. Tucker (eds), *Advances in Game Theory*, Princeton University Press, pp. 211–253.

Zermelo, E. F. (1913). Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels (in German), *Proceedings of the Fifth International Congress of Mathematicians*, Cambridge, UK, pp. 501–504.