Game Theoretic Approaches to Motion Planning in Robot Soccer

von der Fakultät für Elektrotechnik,
Informatik und Mathematik
der Universität Paderborn
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
(Dr. rer. nat.)

genehmigte Dissertation

von
Marcus Post

Paderborn, 2008

Referees: Prof. Dr. Oliver Junge
Prof. Dr. Burkhard Monien

Committee: Prof. Dr. Michael Dellnitz (chairman)
Prof. Dr. Peter Bürgisser
Prof. Dr. Oliver Junge
Prof. Dr. Burkhard Monien
Dr. Elke Wolf

Date of PhD Defense: 17.04.2008

You can not learn anything
until you already almost know it.

Unknown

To Berthild, Karl-Friedrich, Sebastian, and Ping

Acknowledgements

I would like to start by thanking my advisors Prof. Dr. Michael Dellnitz and Prof. Dr. Oliver Junge for their guidance, support, motivation, and for the great freedom which was given to me. Prof. Dellnitz' group at the University of Paderborn always provided a very good research environment for me. I also want to thank Prof. Dr. Burkhard Monien for helpful comments and for reviewing this PhD thesis.
Moreover, I am very grateful to Prof. Dr. Oliver Junge, Prof. Dr. Michael Dellnitz, Dr. Sina Ober-Blöbaum, Dr. Kathrin Padberg, Dr. Oliver Schütze, Stefan Sertl, and Bianca Thiere for interesting discussions and exciting joint work. For fruitful discussions and comments I would also like to thank Mirko Hessel-von Molo, Oliver Kramer, Prof. Dr. Michael G. Lagoudakis, Tim Laue, Dr. Martin Lauer, Henning Meyerhenke, and Willi Richert.
In general, I am indebted to my colleagues in Paderborn including Alessandro Dell' Aere, Sebastian Hage-Packhäuser, Mirko Hessel-von Molo, Stefan Klus, Dr. Arvind Krishnamurthy, Anna-Lena Meier, Dr. Sina Ober-Blöbaum, Dr. Kathrin Padberg, Michael Petry, Dr. Robert Preis, Dr. Oliver Schütze, Stefan Sertl, Bianca Thiere, Dr. Fang Wang, and Katrin Witting for discussions, technical support, social events and many other things. I am further indebted to the collegiates of the PaSCo graduate school.
I am grateful to Laurel Frick-Wright for proofreading my thesis to improve the quality of my English and to Mirko Hessel-von Molo, Anna-Lena Meier, and Stefan Klus for proofreading excerpts.
I always received valuable administrative support from the secretaries Marianne Kalle and Tanja Bürger, and from Anne Belkner. For enabling my work in a different way, namely by keeping my office clean, I thank the non-scientific staff of the University of Paderborn.
I would like to thank Alessandro Dell' Aere for helping me to build a physical soccer environment for the AIBO robots and some students for supporting me with the AIBO robots' technical issues: Johannes Berg, Raphael Golombek, and Nicolai Hähnle are to be mentioned here.
Last but not least I am very indebted to my parents Berthild and Karl-Friedrich Post who supported me not only during my studies but during my whole life in all ways imaginable. I want to thank my brother Sebastian, Janina, my friends from the University of Paderborn, especially Ping, the "Mensa-Kreis", and the musicians I played music with, and, of course, all other friends of mine.
For the development of great software tools I want to especially thank all LaTeX developers and the developers of Dia, Kate, and Kile many of whom work voluntarily, and the developers of MATLAB which is a commercial software tool. All of them made my work technically possible or at least substantially simpler.
For financial support I am very grateful to the Paderborn Institute for Scientific Computation (PaSCo)[1] and to the University of Paderborn.

Marcus Post, February 2008

[1] The research is (partly) supported by the DFG Research Training Group GK-693 of the Paderborn Institute for Scientific Computation (PaSCo).

Abstract

Robotics is, from a scientific point of view, a very broad topic with many applications. While highly specialised robots have been widely used in production lines, the next big scientific steps are towards autonomy of robots and interaction with other robots and humans. For achieving these long-term goals catenations of physical and mental abilities which are interdisciplinary and scientifically challenging have to be carried out. At its current state, robot soccer is an appropriate environment for demonstrating and developing robotic skills as several areas are addressed such as image processing and analysis in the widest sense (including e. g. object matching and directing the camera), control and optimisation of physical movement (walking, ball handling), and the strategic planning which may be considered as being close to high level reasoning. In this thesis, game theoretic and reinforcement learning approaches are utilised to contribute to strategic planning in robot soccer which serves as a motivating example. The aim of strategic planning is to obtain an optimal strategy which also takes the possibly unknown strategies of other players into account. A natural further goal of this thesis is the development and analysis of algorithms by means of which such optimal strategies can be approximately computed.

More specifically, the following steps are undertaken: first, a game theoretic model of multi-player robot soccer is developed which is independent from the robot hardware. The occuring challenge to determine an optimal strategy with respect to this model for as many robots as possible is met by exact model reduction, i. e. by finding equivalent smaller models. For this, a theoretical framework of symmetries is developed which bases on homomorphisms between two-player zero-sum Markov games. It lays a formal foundation for practitioners who already implicitly used results proven within that framework. A special result which is important for model reduction is that the reduction can be performed in several separate steps and be combined afterwards which is expressed by a composition of homomorphisms. Finally, a qualitatively new symmetry which interchanges the two players of the Markov game, i. e. the two teams in robot soccer, is proven to be part of the homomorphism framework. Particularly, this means that it can be combined with all symmetries which occur in Markov decision processes.

The theoretical results about Markov game symmetries are algorithmically exploited for Dynamic Programming (DP) and Reinforcement Learning (RL) methods which are also compared. Such comparisons ought to be standard but seem unusual for large parts of the RL literature. Unsurprisingly, DP methods are more efficient and thus the following general procedure seems recommendable: firstly, to design an approximate model for the task at hand and solve this by DP methods to an appropriate level of precision and, secondly, to use the DP solution of the rough model as an initialisation for an RL method to let the RL method adapt to the unknown real model. In this spirit, the developed soccer model and the computation of its optimal solution can be seen as the completion of the first of the above two steps. Ideas of dynamical systems and graph theory are additionally integrated

to design new efficient DP methods by means of almost invariant sets. All algorithms are thoroughly studied numerically and the results of optimal strategies are also interpreted in terms of soccer. Finally, some of the most challenging future tasks to implement these strategies on real robots are identified.

Key Words

reinforcement learning, robotics, robot soccer, optimal strategy, symmetry, model reduction, control theory, game theory, graph theory, dynamical system, almost invariant set, homomorphism

Abstract (German)

Die Robotik ist aus wissenschaftlicher Sicht ein sehr breites Fachgebiet, das viele Anwendungen hat. Weitverbreitet sind beispielsweise hochspezialisierte Roboter, die in der maschinellen Fertigung eingesetzt werden. Einige der nächsten Meilensteine in der Robotik sind in der Autonomie von Robotern und in der Interaktion mit Robotern und Menschen zu erwarten. Zum Erreichen dieser Meilensteine ist eine Verknüpfung von physischen und "mentalen" Fähigkeiten notwendig, die interdisziplinär ist und wissenschaftliche Herausforderungen bietet. Roboterfußball stellt zum derzeitigen Stand der Wissenschaft eine geeignete Umgebung dar, um verschiedenartige Fähigkeiten der Roboter zu demonstrieren und weiterzuentwickeln, denn es beinhaltet bereits eine Vielzahl von Gebieten, beispielsweise Bildverarbeitung im weitesten Sinne (einschließlich Objekterkennnung und -verfolgung), Kontrolle und Optimierung physischer Bewegung (Fortbewegung, Ballfertigkeiten) und die strategische Planung, die auch als höhere kognitive Fähigkeit betrachtet werden kann. In dieser Doktorarbeit werden Ansätze der Spieltheorie und des Reinforcement-Learnings genutzt, um Beiträge zur strategischen Bewegungsplanung im Roboterfußball, das als motivierendes Beispiel dient, zu leisten. Ziel der Strategieplanung ist es, eine optimale Strategie zu ermitteln, die auch die möglicherweise unbekannten Strategien anderer Spieler einbezieht. Ein weiterführendes Ziel der Arbeit stellt die Weiterentwicklung und Analyse von Algorithmen, mit deren Hilfe optimale Strategien approximativ berechnet werden, dar.

Dazu werden die folgenden Schritte unternommen: Zunächst wird ein spieltheoretisches Modell des Mehrspieler-Roboterfußballs entwickelt, das möglichst hardware-unabhängig ist. Einer wesentlichen dabei auftauchenden Herausforderung, optimale Strategien für dieses Modell mit einer möglichst großen Anzahl von Robotern zu bestimmen, wird durch exakte Modellreduktion begegnet, d. h. es wird versucht, ein möglichst kleines, dem originalen Modell äquivalentes Markov-Spiel zu ermitteln. Zu diesem Zweck wird ein theoretisches Konzept von Symmetrien eingeführt, das auf Homomorphismen zwischen Zweispieler-Nullsummenspielen basiert. Der Symmetriebegriff schafft dabei eine formale Basis für schon zuvor zur praktischen Lösung von Markov-Spielen implizit angewendeten Symmetriereduktionen. Ein nützliches Ergebnis für die Modellreduktion ist, dass diese schrittweise durchgeführt und anschließend kombiniert werden kann, was sich formal durch die Komposition von Homomorphismen darstellen lässt. Schließlich ist eine qualitativ neuartige Symmetrie, die die Spieler eines Markov-Spiels vertauscht, in den Formalismus integriert. Insbesondere wird gezeigt, dass sich die nicht in Markov-Entscheidungsprozessen vorkommende Symmetrie mit allen dort anzutreffenden Symmetrien kombinieren lässt.

Die theoretischen Ergebnisse über Symmetrien in Markov-Spielen werden algorithmisch umgesetzt in Methoden der Dynamischen Programmierung (DP) und des Reinforcement-Learnings (RL), welche ferner miteinander verglichen werden. Derartige Vergleiche sollten als Standard gelten, scheinen aber eher die Ausnahme in weiten Teilen der RL Literatur zu sein. Erwartungsgemäß sind die DP Methoden effizienter, weshalb die folgende allgemeine

Vorgehensweise vorgeschlagen wird: zunächst ein approximatives Modell zu konstruieren und mit Hilfe der DP Methoden zu lösen, um dann ein RL Verfahren mit dieser Lösung als Startwert auszustatten. Dies ermöglicht sowohl den Einsatz der effizienteren DP Methoden, die mit angemessener Präzision das approximative Modell lösen, als auch den der RL Methoden, deren Adaptivität an das unbekannte reale Modell ausgenutzt wird. In diesem Sinne können das entwickelte Roboterfußball-Modell und die Berechnung optimaler Strategien als Lösung des ersten Teils des allgemeinen Vorgehens angesehen werden. Dazu finden bei der Entwicklung neuer effizienter Algorithmen unter anderem Ideen aus dem Gebiet der Dynamischen Systeme und der Graphentheorie zu fast invarianten Mengen Anwendung. Abschließend werden wichtige praktische Herausforderungen identifiziert, die es zu lösen gilt, bevor die berechneten optimalen Strategien auf reale Roboter übertragen werden können.

Schlagworte

Reinforcement-Learning, Robotik, Roboterfußball, optimale Strategie, Symmetrie, Modellreduktion, Kontrolltheorie, Spieltheorie, Graphentheorie, Dynamisches System, fast invariante Menge, Homomorphismus

Contents

Chapter 1

Introduction

One of the most impacting technical revolutions in the near future after the boom of computer technology and the Internet could be expected in robotics. While highly specialised robots have been widely used in production lines the interaction of robots with other robots or humans remains a vision. The catenation of physical and mental abilities of robots is scientifically highly challenging. The physical abilities are often referred to as lower level abilities such as reflexes and basic motion abilities. However, it is not completely clear how much the physical abilities are intertwined with cognitive ones. Imagine, for example, that a human hits an obstacle by a physical movement. A negative reward signal is received in the form of pain which causes the cognitive section to think how to avoid that pain. This high level cognitive activity can at first be concentrated on the special situation but then be generalised to a set of similar situations and at the end possibly result in an improvement of the basic ability of moving, e. g. if it is generally sensible to move more slowly.

Key aspects of the present work are informally described and related to the introductionary example above. In this thesis, optimal strategies of several robots which rule their interaction are determined for the example of robot soccer. The highest cognitive level of strategic planning is addressed. The above example of a human hitting an obstacle already points to important ingredients of a mathematical description and solution methods: firstly, the physical movement indicates that a dynamic model of an environment and some acting agents is needed. Secondly, a kind of optimality criterion has to be employed based on a local reward signal (pain) which could also be positive, and thirdly, after receiving any reward an estimate of how to optimally act to maximise a positive reward while avoiding a negative one has to be improved. A basic principle for algorithms seems to be that some estimates of how useful a special situation is have to be improved over time. These estimates are later represented by a value function and the iterative procedures are refered as dynamic programming (DP) and reinforcement learning (RL) methods. Finally, the ability to abstract former knowledge of how to deal with similar situations which have not been explicitly experienced appears to be crucial. This question is theoretically addressed by developing a framework of exact model reduction, i. e. identifying *exactly* equal situations, as well as practically by incorporating *approximate* generalisation methods which are introduced as supervised learning methods.

A huge variety of concepts and algorithms to which the above mentioned aspects belong also are collectively named *machine learning*. In the remainder of this introduction an overview of different topics is given: goals of this thesis are presented in more detail and several disciplines of machine learning relevant to this thesis are introduced. The areas of

supervised learning (function approximation) and reinforcement learning (optimal control) are touched on. In addition, relations to game theory and to the concept of almost invariant sets established in dynamical systems theory are shortly pointed out. Finally, challenges of implementing optimal strategies on real soccer playing robots are discussed before the introduction ends with the contributions of the author.

Goals of the Present Thesis. Since in this thesis optimal strategies for motion planning in robot soccer are to be obtained the first goal is to design an appropriate soccer model which is as simple as possible and as special as necessary. The simplicity aims at wide applicability whereas a certain level of detail is necessary to provide meaningful results.

A more general goal is to provide and analyse efficient algorithms for computing optimal strategies. Such algorithms are applicable independently from the model as long as it belongs to a certain but broad class. A natural small subtask is the calculation of optimal strategies especially for the robot soccer model, i. e. to give for each soccer situation (state $s \in \mathcal{S}$) a probability distribution of which action a of the action space $\mathcal{A}(s)$ to perform. This subtask gives reason to define the development of a software package which implements all key ideas as a further substantial goal.

Beyond designing efficient algorithms, the intent to widen applicability of the algorithms shall also be pursued by reducing models to equivalent smaller ones. In this area, the focus is on symmetry reduction which has to be appropriately defined for two-player zero-sum Markov games. Furthermore, the proof of desired properties e. g. which consequences the symmetry of a model has for the symmetry of optimal strategies is an important goal of this thesis.

Machine Learning, Game Theory, and Dynamical Systems

Although often a different impression could be received machine learning [139] and stochastic control are intertwined, as e. g. [10, 145] discuss the relevance of control theory topics to Artificial Intelligence (AI). In stochastic control, adaptive control is the field concerned with algorithms which improve a sequence of decisions from experience. It is a mature discipline for systems with smooth dynamics, see e. g. [26, 195].[(1)] In contrast, learning methods are often applied to time-discrete systems and utilise special function approximation (supervised learning) or data preprocessing (unsupervised learning) methods. However, the most decisive difference is the knowledge of the underlying model: in learning methods the model can be completely unknown whereas in control theory the model typically is assumed to be known.[(2)] Throughout this thesis, the author has chosen the machine learning terminology. This should not be misunderstood as an evaluation of both approaches.

Machine Learning. There are three major types of learning (Figure 1.1) with different degrees of freedom for the learner: completely *supervised learning* (SL, Chapter 4), completely *unsupervised learning*, and lying between these two extremes *reinforcement learning* (RL, Chapter 2). In several applications these different types of learning are intertwined, for example [193] gives a short overview of combinations of supervised, unsupervised and reinforcement learning.

Supervised learning methods are often interpreted as generalisation methods and deal with function approximation for a given set of samples. For example, let $f : \mathbb{R}^n \to \mathbb{R}$ be a

[(1)] In [195] a good overview of numerical methods in optimal control can also be found.
[(2)] By addition of system identification methods this distinction can also become blurred.

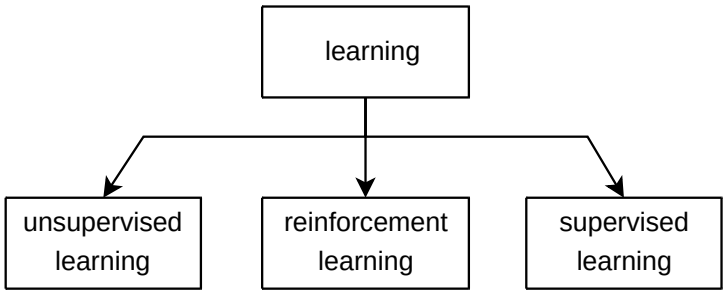


Figure 1.1: Scheme of the main subfields of learning.

function and $S_{\text{input}} = \{s_1, \ldots, s_m\} \subseteq \mathbb{R}^n$ a finite subset of the domain of f. The set of input-output pairs $S = \{(s_k, f(s_k))\}_{k=1,\ldots,m}$ is called the sample set. Then, the task of SL methods is to find $\widetilde{f} \approx f$, i. e. minimising the norm difference with respect to a given norm, by only using information of the sample set. A popular choice of $\widetilde{f}$ is a linear approximation architecture $\widetilde{f} = \sum_j a_j f_j$ with specified f_i and a_i to determine. In contrast to function approximation methods some SL methods deal with noise in the process of function evaluation, i. e. $f(s_i)$ is deviating from the assumed value in the input-output pair. Furthermore, an SL specific topic is feature detection which is related to a partly automatic design of the f_i. The use of handcrafted features is computationally less demanding which is subject to numerical studies in Chapter 5 and yielded reasonable results even if only simple features are used.

Unsupervised learning methods are also seen as generalisation methods like SL but they differ by only making use of the distribution of input data S_{input}. Thus, they can be interpreted as input data preprocessing methods to which e. g. clustering approaches belong to.

Reinforcement learning methods do not directly fit to the concept of the above two methods. They are biologically inspired by action-reward animal training mechanisms. The mathematical foundations are based on a combination of optimal control and stochastic approximation methods. The basic ingredients are a state and an action space, a dynamical probabilistic model called transition function according to a specified time set (discrete or continuous), and a reward function which evaluates every action in each situation and is typically non-zero only in a few situations. The goal is to determine a behaviour that ensures a high long-term reward which includes for example the tedious work to figure out which situations are the most positive ones and how these can be achieved most effectively. Effectivity is typically measured by rewards and time (time discounting of reward). One of the most applied equations is the Bellman equation (Equations 2.48 and 2.49, [66]) which gives a rule to propagate the estimations of usefulness of each situation one step further in time. Many iterative methods for computing optimal strategies such as value iteration and Q-learning are based on the Bellman equation.

A general discussion about the use of learning paired with some criticism about current research and an excellent overview of multi-agent learning can be found in [188].

Game Theory. In reinforcement learning approaches it is the goal to determine an optimal behaviour according to some optimisation criterion. This behaviour is usually interpreted as the active decision of an agent and, thus, it is natural to consider models with several agents. This path directly leads to game theory. While game theory is often influenced by economic flavours the basic question of game theory of how to act optimally

in an environment with several players, respectively agents, is quite general. The equally general answer is to find a Nash equilibrium. However, special foci of classical game theory are rivalry or conflicting aims of several players, coordination and coalitions, threat mechanisms, partial information, games of social welfare, and so on. From a computational point of view finding Nash equilibria can become arbitrarily hard for games with more than two agents [41].[(3)] Fortunately, in robot soccer the different agents of each team can be represented by only one game theoretic agent because the team is fully cooperative. Furthermore, the game is of type "zero-sum" which can be interpreted as games between two rivals: one agent wants to avoid what the other tries to achieve and vice versa. Sometimes such a situation is called (completely) *competitive* because there is no potential for a compromise. Typically, board games (backgammon, chess) or sport games (soccer, tennis) are of this type because one agent or team of agents wins if and only if the other one loses.

Connections to Dynamical Systems and Graph Theory: Almost Invariant Sets.

The update steps of *synchronous* dynamic programming (DP) algorithms are global, while the updates of reinforcement learning methods are local with respect to specific stochastic trajectories. To combine these two concepts, almost invariant sets are employed. These characterise regions which are left by a stochastic trajectory with low probability, and thus give information about where a typical reinforcement learning trajectory will stay for a long time. In Section 5.3, it is shown how to exploit this knowledge to design an *asynchronous* DP algorithm for which flexible trade-offs between global and local updates are possible.

The idea is to utilise an estimate of the Nash equilibrium strategies to analyse the dynamics of the Markov game. If all players' strategies are fixed such that the dynamics do not explicitly depend on the strategies, the concept is identical to that of analysing the dynamics of dynamical systems by means of discretised transfer operators [46]. By the invariance information of the dynamics regions for repeated local updates, namely the almost invariant sets, can be located. Finally, it seems algorithmically sensible to alter global and local update steps. For computing a partition of almost invariant sets *graph partitioning* techniques are utilised.

The invariance information could have a two-fold impact: it may directly yield valuable information to speed up numerical algorithms (e. g. value iteration) and may give – on a meta-level – useful information about which situations in robot soccer are dynamically linked if both teams are playing optimally.[(4)] An application could be to exploit suboptimal behaviour of the other team to perform a controlled jump from a nearly invariant component to another more advantageous one.

Robot Soccer with Real Robots

Multi-player robot soccer is the main application as well as the standard example to illustrate theoretical concepts throughout this thesis. The model used for numerical computations is described in detail in Section 5.1. Here, only the most salient aspects are mentioned. Since the model addresses the highest level planning and does not resolve lower levels of behaviour such as direct motor controls of the robots, it is possible to design a model that is widely applicable. Particularly, the proposed model is relatively hardware independent. Basic abilities such as walking towards a predefined location, handling the

[(3)] Communicated by Prof. Dr. Bernd Sturmfels in a mathematical colloquium of the University of Paderborn.

[(4)] These could be called a *meta stable* set of situations.

ball by dribbling and kicking, and skills for the analysis of visual information (self and opponent localisation) are highly non-trivial but assumed to be already available. The only assumption about the abilities of the robots is qualitative, namely that both teams including every single robot are totally equal. This is true e. g. for the AIBO league[(5)] but not for all leagues of robot soccer. The consequences of equality of all robots are that the robots are undistinguishable and that it is at least possible for each robot to apply the same basic abilities. The question of whether simulation results can be transferred to real robots is answered positively by [69]. This shows that it can be valuable to compute an optimal strategy for a non-exact model (simulation).

Reading Information and Contributions of this Thesis

In this thesis a game theoretic model of robot soccer and new algorithms for obtaining optimal strategies within this model are developed. This results in the following structure of the thesis, the description of which emphasises the contributions of the author:

A large portion of **Chapter 2** is a special collection of common knowledge about Markov decision processes (MDPs), reinforcement learning, and different types of games. The interpretation of matrix games (Definition 2.18) in the more general context of two-player zero-sum Markov games (2P-ZS-MGs) and the numerical error analysis of the value iteration for 2P-ZS-MGs (Lemma 2.36) are contributed by the author. The latter seems to be necessary because the solution of matrix games in the iteration step introduces numerical errors which lead to a modified stopping criterion of the iteration scheme. The analysis is mathematically identical to the error analysis of using supervised learning methods in Chapter 4 but the consequences are different: for standard value iteration the total error is typically only corrected a little whereas for function approximation the same error analysis reveals that no guarantee of the quality of the solution can be given.

The analysis is originally developed in the context of Chapter 4 for the theoretically identical case of function approximation but the assumptions and consequences are different: in the numerical value iteration the total error is typically just corrected a little whereas for function approximation the same error analysis reveals that it destroys any guarantees on the quality of the solution.

Chapter 3 deals with model reduction. Section 3.1 begins with an introduction of the concepts of MDP homomorphisms and MDP symmetries. By a new and algebraicly more precise way of presentation the relation between concepts of different authors can be clarified. A key result of the author is to show that the formalisms of MDP homomorphisms [171] and MDP symmetries [217] are equivalent (Lemma 3.3). This equivalence reveals that the model minimisation framework of [171] with MDP options and the symmetry context of [217] with its generalisation to multi-agent MDPs does not exclude each other but are based on the same foundation. Additionally, the concept of the symmetry group of MDPs by [171] is related to its natural algebraic framework of group actions by the author.

A second key result is the development of a corresponding framework for the case of 2P-ZS-MGs including all basic statements equivalent to those for MDPs in Section 3.2. This more general framework was necessary to capture the symmetries of the robot soccer model

[(5)] The current development of the four legged league was influenced by the fact that the production of the AIBO ERS-7 was stopped recently. The new name of the league is Standard Platform League (SPL) meaning that all robots have to be equal (as before) but additionally a new two legged humanoid robot called NAO is introduced as a substitute for the AIBO "dog".

developed in Chapter 5. It can be expected that 2P-ZS-MGs are one of the most general types of models which include MDPs and for which such statements are true.

Besides the symmetries of MDPs, a qualitatively new symmetry that results from exchanging the two players can be exploited. Practically, such symmetries have been used within different board games by the argument that exchanging the players in a zero-sum game has to result in a multiplication of the value by -1. One example is given by the pioneer Samuel for checkers [179]. The present work lays a formal foundation for this argument and, more importantly, shows that this exchange of players is also compatible with the other standard symmetries similar to that of MDPs (Proposition 3.24) and that symmetry reduction can be performed stepwise.

The key result of **Chapter 4** which is devoted to the combination of reinforcement learning and supervised learning is Lemma 4.2 and its implications. It clarifies that unless a function approximation architecture is very accurate care is needed if the quality of the solutions is to be provable. Nevertheless, a collection of successfully applied results of other authors is provided. Corresponding results of the author can be found in Section 5.2.6.

Finally, **Chapter 5** provides detailed information about the model and the numerical results. The multi-player grid soccer model described in Section 5.1 is based on [112] but goes far beyond: the generalisation to several agents per team makes it necessary to change from a one-agent "blocking dynamic" to a more permeable one and allows it to introduce passes between different agents of the same team. Furthermore, the author believes that the model is very well-suited to the study of multi-agent systems: since the grid size can be varied as well as the number of agents, effects of large state spaces can comparatively be studied, introduced by both high grid resolution and by multiple agents on a low resolution grid.

Section 5.2 contains the numerical results and possesses a rich substructure. Some of the studies are providing strong arguments for the choice of 2P-ZS-MG instead of MDP models in robot soccer (Section 5.2.2), others compare dynamic programming and reinforcement learning techniques (Section 5.2.4) with the result that exact and model exploiting dynamic programming techniques should be preferred whenever possible. After these evaluative numerical results which appear to be natural but are surprisingly not standard in literature the studies of dynamic programming methods are intensified in Section 5.2.5 and the dependency of convergence speed on different types of methods and parameters is numerically analysed. This includes comparative studies of symmetry reduced models with their unreduced counterparts which is the practical application of the theoretical results in Chapter 3. Particularly interesting from a practical point of view is the "max-min convergence boosting phenomenon" which only seems to be present in 2P-ZS-MGs but not in MDPs. As mentioned above, results with supervised learning (Section 5.2.6) and general technical issues of applying strategies to real robots (Section 5.4)[(6)], especially of type AIBO ERS-7, follow.

Chapter 6 concludes and points to future work and the **appendix** contains a short description of group actions (Appendix A), comments on the relations of the iterative solution of the Bellman equation to iterative linear solvers (Appendix B), an introduction to the software package DRPOST developed by the author (Appendix C), and additional material (Appendix D) omitted in the main part.

[(6)] These technical issues were not discovered by the author but independently confirmed.

Chapter 2

Reinforcement Learning (RL) and Game Theory

Contents

> The goal of machine learning is to design a software
> that has more abilities than its programmer in the end.
>
> INSPIRED BY ARTHUR L. SAMUEL (1901-1990)

A very general framework to which this thesis contributes is that of *multi-agent systems* (MASs) [64] and more specifically to multi-agent learning [188]. To cut a long story short, the basic ingredients of MASs are a number of *agents*[(1)] which *interact* with an *environ-*

[(1)] Agents are not restricted to physical agents like humans or robots. Reversely, if a robot is not capable of making decisions then it is part of the environment and does not count as an agent.

ment and *perceive* some information from the environment. The role and (hierarchical) structures of agents are also discussed in the MAS context.

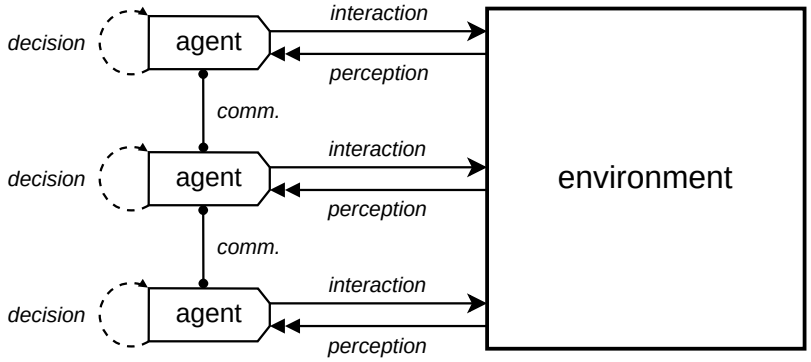


Figure 2.1: Scheme of the main components of multi-agent systems. Multiple agents interact with an environment and partially perceive information from the environment. The agents are allowed to communicate and make some decisions for their future interactions.

Bowling comments the usefulness of the MAS context being a general framework as follows [21]:

> Frameworks are models of reality. As such, they are an important foundation for the generation and evaluation of new ideas. They establish the rules of the game, crystallise the core issues, provide a common basis of study, make intrinsic assumptions visible, provide a general perspective on large classes of problems, help to categorise the variety of solutions, and allow comparison with other models of reality. It is with all of these reasons in mind that we begin this work by introducing a framework for multiagent learning.

The class of *stochastic games* is a more specific framework than the general MASs.[(2)] These define the guidelines of this work and are rarely used for reinforcement learning tasks – exceptions are [21, 98, 112] the first of which also inspired the structure of this section. Stochastic games are a generalisation of Markov decision processes (MDPs) and matrix games. The first is the research focus of reinforcement learning (RL) and dynamic programming (DP), the second is a major topic of game theory with applications in economic science. In the remainder of this section, basic ingredients are defined and some well-known results about the solution of basic problems are presented.

More precisely, this section is organised by increasing model complexity as follows: First, Section 2.1 starts with a brief overview of *dynamical systems* which provide the framework for systems without any decision maker. Nevertheless, fixing the policies of all agents of an MAS results in a (stochastic) dynamical system and hence some basic insights from this mathematical discipline may serve as inspiration for Markov games. Even more of interest, a *deterministic continuous* dynamical system can also be approximated by a *stochastic discrete* dynamical system [46] – or in other words an MDP without a decision maker[(3)]. We proceed by introducing MDPs in Section 2.2 (one decision maker), matrix games (two adversary decision makers) in Section 2.3, and two-player zero-sum Markov games in Section 2.4 (also two adversary decision makers). In the last section, an overview of more advanced concepts which are only partly relevant to this work is given. This

[(2)] An alternative (exhaustive) framework for continuous motion planning problems can be found in [104]. However, practical application seems limited at the moment to special cases.

[(3)] A stochastic discrete dynamical system is often called *Markov process* or *Markov chain*.

includes semi-Markov decision processes (S-MDPs) and hierarchical approaches. In each subsection aspects of history, notation, and the basic problems are provided. Summarising, this chapter is the foundation of this thesis.

2.1 Dynamical Systems and Markov Processes

Dynamical systems are a very general concept and include concepts for discrete and continuous systems as well as for deterministic and stochastic ones which are also called random dynamical systems. Many concepts were developed to analyse emerging properties of deterministic dynamical systems such as equilibrium and periodical solutions, symbolic dynamics, chaos, and relations between numerics and dynamical systems. A special goal of dynamical systems theory is to provide statements about the asymptotic long-term behaviour of a given system. Additionally, as will be pointed out in the following, extracting statistical information of (deterministic) dynamical systems is interesting for many systems and leads to relations of deterministic continuous dynamical systems and stochastic discrete ones (see also footnote **(3)**) by so-called transfer operators.

As mentioned above, dynamical systems are considered in the beginning of Chapter 2 because they describe situations where *no* agent, decision maker, or controller is involved. These three terms are synonymously used by different communities. However, a (stochastic) dynamical system arises if one considers the policies of all agents of an MAS being fixed. The ideas of the following short overview are mainly borrowed from [46]. In addition to this article, a more detailed description and further references of approximating deterministic dynamical systems by Markov processes can be found in [48, 49, 50]. A less specific introduction to the broad area of dynamical systems may be obtained by reading the textbooks [25, 73, 153] the last of which contains many examples.

Historical Remarks. Following [153], early work of dynamical systems was motivated by the aim of predicting the non-linear dynamics of the n-body problem in celestial mechanics. A pioneer in this field was Poincaré who used perturbation methods and employed a geometrical qualitative point of view. Further major contributions were achieved by Arnold, Birkhoff, Bowen, Duffing, Kolmogorov, Krylov, Lorenz, Lyapunov, Moser, Pontryagin, Rayleigh, Ruelle, Smale, and many others. Cartwright and Littlewood [28] observed *chaos* which was also discovered by Lorenz studying weather dynamics [117] and separately by Smale who introduced the horseshoe map [192].

2.1.1 Basics and Problem Definitions

After a short introduction to deterministic dynamical systems and invariant sets Markov processes and some concepts to approximate the statistics of dynamical systems by means of transfer operators will be introduced.

Deterministic Dynamical Systems

The first definition of dynamical systems seems very abstract but later more specific aspects will be discussed.

2.1 Definition (Dynamical System [25])
Let X be a non-empty set and $\{\varphi^t : X \to X \mid t \in \mathbb{K}\}$ a one-parametric family of maps with parameter t and $\mathbb{K} = \mathbb{R}$ or $\mathbb{K} = Z$. If this family is a one-parametric group, that means

$\varphi^{t+s} = \varphi^t \circ \varphi^s$ *and* $\varphi^0 = \mathrm{id}_X$ *is the identity map, then* (X, φ) *is called a time-continuous (autonomous) dynamical system if* $\mathbb{K} = \mathbb{R}$ *and a time-discrete one if* $\mathbb{K} = \mathbb{Z}$. φ *is the* flow of the dynamical system.

In an irreversible system the time may be restricted to $t \geq 0$ resulting in a semi group instead of a group structure. For a dynamical system (X, φ) and $x \in X$ the set $\mathcal{O}_\varphi^+(x) = \bigcup_{t \geq 0} \varphi^t(x)$ is called the *positive semi orbit*, $\mathcal{O}_\varphi^-(x) = \bigcup_{t \leq 0} \varphi^t(x)$ the *negative semi orbit* and $\mathcal{O}_\varphi(x) = \mathcal{O}_\varphi^+ \cup \mathcal{O}_\varphi^- = \bigcup_t \varphi^t(x)$ the *orbit* or *trajectory* of x under flow φ. If the context is clear then φ may be omitted.

2.2 Definition (*Invariant Set*)
The set $A \subseteq X$ *is called* φ*-invariant if* $\forall t : \varphi^t(A) \subseteq A$. *If a dynamical system is not time-invertible then* $\varphi^{-t}(A) = (\varphi^t)^{-1}(A) = \{x \in X : \varphi^t(x) \in A\}$ *is the set of preimages of* A *under* φ^t. *A* forward invariant set *possesses the invariance property for all* $t \geq 0$, *a* backward invariant set *for all* $t \leq 0$.

Dynamical systems and differential equations. According to [25], flows of dynamical systems occur naturally in autonomous differential equations of the first order. That means equations of the form

$$\dot{x} = f(x), \tag{2.1}$$

where $x \in X = \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}^n$ is a continuous differentiable function ($f \in C^1$), and $\dot{x} = \frac{\mathrm{d}x}{\mathrm{d}t}$. For each $x \in X$ there exists a unique solution $\varphi^t(x)$ with $\varphi^0(x) = x$ which is defined for all $t \in U_\varepsilon(0)$ in a neighbourhood of 0. If the solution exists for all t (which is assumed in the following and is true e. g. for bounded f) then for fixed t a C^1-diffeomorphism of $\mathbb{R}^n$ is given by the map $x \mapsto \varphi^t(x)$. Because the differential equation was autonomous it holds $\varphi^{t+s} = \varphi^t(\varphi^s(x))$ meaning that φ^t is a flow. Reversely, for a given differentiable flow $\varphi^t : \mathbb{R}^n \to \mathbb{R}^n$ the corresponding differential equation reads to

$$\dot{x} = \frac{\mathrm{d}}{\mathrm{d}t} \varphi^t(x)\Big|_{t=0} \tag{2.2}$$

because if equation 2.1 is integrable a solution $x(t)$ fulfills: $\varphi^t(x(0)) = x(t) = x(0) + \int_0^t f(x(s))\,\mathrm{d}s$.

Transfer Operators, Almost Invariant Sets, and Markov Processes

This paragraph is based on [46] and deals with time-discrete dynamical systems of the form

$$x_{k+1} = f(x_k) \tag{2.3}$$

with $k \in \mathbb{N}_0$ and $f : X \to X$ for a compact X are considered. An important class of examples are the time-T maps (for fixed $T \in \mathbb{R}$) of a time-continuous dynamical system (Equation 2.1). If the *global* dynamical behaviour of a given system is of interest then a useful method is to employ the transfer operator or Perron-Frobenius operator associated with f. It describes the evolution of signed measures on X.

The *transfer operator* or *Perron-Frobenius operator* of a time-discrete dynamical system f is the linear operator $P : \mathcal{M} \to \mathcal{M}$ defined for all measurable sets S by

$$(P\nu)(S) = \nu(f^{-1}(S)) \tag{2.4}$$

where $\mathcal{M}$ is the space of signed measures on the Borel σ-algebra over X. For example, if ν is a uniformly distributed probability measure on a set $S_1 \subseteq X$ (meaning uniformly distributed on S_1 and 0 elsewhere) then $(P\nu)(S_2)$ is exactly the transition probability $T(S_1, S_2)$ of Definition 2.5 because $\nu(S_1) = 1$. In contrast to the transition function T of Section 2.2, which maps only states and actions to probabilities, the transfer operator P directly maps a probability distribution of inputs (states at time t) to a probability distribution of outputs (states at time $t+1$). In the context of dynamical systems the case of a measure μ, which is a fixed point of the transfer operator, is considered, i. e. μ is invariant with respect to f. Then for all measurable sets S the following holds: $\mu(S) = \mu(f^{-1}(S)) = (P\mu)(S)$. Furthermore, an additional standard assumption in this context is that the measure is robust under small random perturbations leading to a unique SRB measure (Sinai, Ruelle, Bowen).

The next step to Markov decision processes (MDPs) and to almost invariant sets is to define the *transition probability* T from a measurable set S_1 with $\mu(S_1) \neq 0$ to a second measurable set S_2 by

$$T(S_1, S_2) = \frac{\mu(S_1 \cap f^{-1}(S_2))}{\mu(S_1)}\,. \tag{2.5}$$

The transition probability from a set S_1 with non-zero measure into itself is called the *invariance ratio* $T_{\text{inv}}(S_1) = T(S_1, S_1)$. If $T_{\text{inv}}(S_1) \geq 1-\varepsilon$ for $\varepsilon \in [0,1]$ the set S_1 is called $(1-\varepsilon)$-invariant which for $\varepsilon = 0$ turns into pure invariance as in Definition 2.2 – neglecting sets of measure zero. If the exact value of ε is not in the focus of interest the imprecise term *almost invariant set* will be used. Particularly interesting – although elementarily obtainable – is the fact that for f-invariant μ the complement of an $(1-\varepsilon)$-almost invariant set is almost invariant with invariance ratio

$$T_{\text{inv}}(X \setminus S) \geq \left(1 - \varepsilon \cdot \frac{\mu(S)}{\mu(X \setminus S)}\right). \tag{2.6}$$

If $\mu(S) \leq \frac{1}{2}\mu(X)$ then the invariance of the complement $T_{\text{inv}}(X \setminus S) \geq 1-\varepsilon$ i. e. the complementary set is also at least $(1-\varepsilon)$-invariant. This motivates a successive algorithmic way of hierarchically splitting up almost invariant sets into almost invariant subsets, and hence iteratively constructing a partition. A *partition* $\mathcal{P}(X)$ of the state space X of a dynamical system (or any arbitrary set X) is a finite collection of pairwise disjoint subsets $P_i \subseteq X$, $i \leq N \in \mathbb{N}$, with $\mu(P_i) > 0$ and the *covering property* $X = \bigcup_{i=1}^{N} P_i$. A partition $\mathcal{P}(X) = \{P_1, \ldots, P_N\}$ is called $(1-\varepsilon)$-invariant if

$$\min_{i=1,\ldots,N} T_{\text{inv}}(P_i) \geq 1 - \varepsilon\,. \tag{2.7}$$

In contrast to other types of decomposition (e. g. ergodic components) the decomposition of a state space X of a dynamical system into a partition of almost invariant sets is not unique. The reason is that slightly changing a partition of almost invariant sets will typically result in only a minor change of the invariance ratios of the corresponding sets.

2.3 Problem (Partitioning into Almost Invariant Sets)
For a dynamical system (X, φ) and fixed $N \in \mathbb{N}$ find a partition $\mathcal{P}(X) = \{P_1, \ldots, P_N\}$ that maximises the average invariance

$$T_{\text{inv}}(\mathcal{P}(X)) = \frac{1}{N} \sum_{i=1}^{N} T_{\text{inv}}(P_i)\,. \tag{2.8}$$

Relation to Markov chains, MDPs, and RL

A Markov chain with a finite amount of states can be obtained by choosing a finite partition[4] $\mathcal{P}(X) = \{P_1, \dots, P_N\}$ and defining transition probabilities according to $T(P_i, P_j)$. This Markov chain can also be considered as an MDP with a finite amount of states where a Markovian strategy of the decision maker is fixed.

In Section 5.3 almost invariant sets build a bridge from the classical dynamic programming algorithms to the reinforcement learning algorithms by combining the global nature of the first methods with the locality of the second ones. The idea is to perform only a few updates on the whole state space and then to restrict the updates of dynamic programming methods to sets with a high invariance ratio[5] (under some fixed strategy of the decision makers), and to again perform a few global updates and so on.

2.1.2 Numerical Methods

A set oriented *numerical* approach to compute partitions $\mathcal{P}(X)$ which preferably consists of almost invariant sets is described below. The presentation includes the numerical approximation of sets in general, an approximation of the transfer operator, some different versions of partitioning problems, and their transformation to a graph based context.

Discretisation of Transfer Operators

To deal with transfer operators in a numerical way a finite dimensional discretisation of the operator must be introduced. As mentioned above, more detailed information is available e. g. in [46, 48, 49, 50]. The basic idea is to first approximate the state space X by a hierarchically constructed sequence of generalised rectangles (a d-dimensional *box*) and then define the transfer operator by the transition probabilities between these boxes.

One numerical scheme for approximating a space X by a finite collection of boxes is the so-called *subdivision algorithm* [47] and works as follows:

Starting with an initial box $B_0 \subseteq X$, a sequence of *collections of boxes* $(\mathcal{B}_i)_{i \in \mathbb{N}_0}$ with $\mathcal{B}_0 = \{B_0\}$ is constructed by repeating the following two steps iteratively until some stopping criterion is fulfilled. For the $(i+1)$-th iteration this reads to:

1.) Refinement: Given $\mathcal{B}_i$, construct a finer collection of boxes $\widetilde{\mathcal{B}}_{i+1}$ by subdividing each box $B \in \mathcal{B}_i$ along one coordinate axis into two halves.

2.) Selection: Given $\widetilde{\mathcal{B}}_{i+1}$, construct $\mathcal{B}_{i+1}$ by keeping all boxes which intersect with the state space X, i. e. $\mathcal{B}_{i+1} = \{B \in \widetilde{\mathcal{B}}_{i+1} : B \cap X \neq \emptyset\}$.

Each of the two steps entails a consideration of how to perform it numerically. The first step can be done directly or be modified by selecting only parts of all boxes for subdivision. Whether the second step is complicated or not depends on the set X and its mathematical description: in general, i. e. for all X, it can not be expected that the condition $B \cap X \neq \emptyset$ can be exactly *and* efficiently be checked.[6] Finally, after i_{end} steps a stopping criterion is fulfilled, and the above algorithm's output is a finite collection of boxes $\mathcal{B}_{i_{\text{end}}} =$

[4] This partition may be much finer than the almost invariant sets.

[5] Neglecting the evolution of the strategy, an RL trajectotry would typically stay in an almost invariant set for a long time.

[6] For example, for a choice of representative *test points* p_k on the box boundary, a test if $p_k \in X$ can be performed.

$\{B_{1,\,i_{\text{end}}},\, B_{2,\,i_{\text{end}}}, \ldots, B_{N,\,i_{\text{end}}}\}$. These fulfill the following criteria: $X \subseteq \bigcup_{k=1}^{N} B_{k,\,i_{\text{end}}}$ and for all $k \neq l$ holds $\mu_{\text{Leb}}(B_{k,\,i_{\text{end}}} \cap B_{l,\,i_{\text{end}}}) = 0$ where μ_{Leb} is the Lebesgue measure.

For the approximation of the transfer operator we simplify notation by writing B_k instead of $B_{k,\,i_{\text{end}}}$). Hence, the starting point is now a collection of boxes $\mathcal{B} = \{B_1, \ldots, B_N\}$ which approximate the set X e. g. in the sense that some stopping criterion of the subdivision algorithm is fulfilled. The *discretisation of the transfer operator* is carried into execution by means of replacing the space of signed measures $\mathcal{M}$ by $\mathcal{M}_{\mathcal{B}}$ which is the finite-dimensional space of signed measures on the Borel σ-algebra generated by the collection of boxes $\mathcal{B}$. If the collection $\mathcal{B}$ forms a partition (i. e. pairwise intersections being empty instead of being of zero measure) then the generated σ-Algebra contains only arbitrary unions of boxes $B_k \in \mathcal{B}$.[7] A standard basis for the vector space $\mathcal{M}_{\mathcal{B}}$ is the set of N measures which uniformly assign the measure 1 to one fixed box $B_i \in \mathcal{B}$ and 0 to all other boxes. The transfer operator with respect to this basis $P_{\mathcal{B}} : \mathcal{M}_{\mathcal{B}} \to \mathcal{M}_{\mathcal{B}}$ is given by the following matrix of transition probabilities:[8]

$$P_{\mathcal{B}} = (p_{ij})_{ij} = \left(\frac{\mu_{\text{Leb}}(B_j \cap f^{-1}(B_i))}{\mu_{\text{Leb}}(B_j)} \right)_{1 \leq i \leq N,\, 1 \leq j \leq N} \in \mathbb{R}^{N,N}\,. \tag{2.9}$$

The measure of a box is straightforward to compute, however, the measure of intersection has to be approximated by numerical methods. [45] describes some possibilities while a wide-spread method is the so-called *Monte Carlo approach* which is described by [85] and means

$$\mu_{\text{Leb}}(B_j \cap f^{-1}(B_i)) \approx \frac{1}{K} \sum_{k=1}^{K} \chi_{B_i}(f(x_k)) \tag{2.10}$$

with x_k being uniformly selected at random from B_j. This simply means that one has to select K random points in B_j and check whether the image $f(x_k)$ is in B_i. There are efficient numerical techniques for performing this task [48, 47].[9]

2.4 Problem (Discretised Partitioning into Almost Invariant Sets (I))
For a dynamical system (X, φ), a set of boxes $\mathcal{B} = \{B_1, \ldots, B_L\}$, and fixed $N \in \mathbb{N}$ find a partition of boxes $\mathcal{P}(X) = \{P_1, \ldots, P_N\}$ (i. e. for each i there exists an index set $K_i \subseteq \mathbb{N}$ such that $\{K_1, \ldots, K_N\}$ forms a partition of $\{1, \ldots, L\}$, and $P_i = \bigcup_{k \in K_i} B_k$ with each $B_k \in \mathcal{B}$) that maximises the average invariance

$$T_{\text{inv}}(\mathcal{P}(X)) = \frac{1}{N} \sum_{i=1}^{N} T_{\text{inv}}(P_i) = \frac{1}{N} \sum_{i=1}^{N} \frac{\sum_{B_i,\, B_j \subseteq P_k} \mu_{\text{Leb}}(B_j) \cdot p_{ij}}{\sum_{B_j \subseteq P_k} \mu_{\text{Leb}}(B_j)}\,. \tag{2.11}$$

Interpretation in Terms of Graphs

As pointed out in [46] Problem 2.4 can be interpreted as finding an optimal cut of a graph. Given a dynamical system (X, φ) and a partition of the state space by boxes $\mathcal{B} = \mathcal{P}(X)$[10],

[7] To handle intersections of measure zero one may define equivalence classes and equality by identifying all sets of zero measure with the empty set.

[8] p_{ij} is the transition probability from box B_j to box B_i.

[9] The fixed point of P describing the f-invariant measure can be obtained in a discretised version by the eigenvector of $P_{\mathcal{B}}$ to the eigenvalue 1.

[10] If X can not be written as a finite union of boxes, the construction of a box approximation of X as described above may be necessary.

one defines the *transition graph* $G = (V, E)$ as a weighted directed *graph* with *vertex set* $V = \mathcal{B}$ and *edge set*

$$E = \{(B_1, B_2) \in \mathcal{B} \times \mathcal{B} : B_1 \cap f^{-1}(B_2) \neq \emptyset\}\,. \tag{2.12}$$

The condition $f(B_1) \cap B_2 \neq \emptyset$ is equivalent to $B_1 \cap f^{-1}(B_2) \neq \emptyset$ even if f is not invertible (bijective). The function of *vertex weights* is defined by

$$v : V \to \mathbb{R},\ v(B_i) = \mu_{\mathrm{Leb}}(B_i)\,, \tag{2.13}$$

and the function of *edge weights* by

$$e : E \to \mathbb{R},\ e(\,(B_i, B_j)\,) = \mu_{\mathrm{Leb}}(B_i) \cdot p_{ji}\,. \tag{2.14}$$

There also exists an *undirected version of the transition graph*, namely $\widetilde{G} = (V, \widetilde{E})$ with the modified edge set

$$\widetilde{E} = \{(B_1, B_2) \in \mathcal{B} \times \mathcal{B} : (B_1 \cap f^{-1}(B_2)) \ \cup\ (B_2 \cap f^{-1}(B_1)) \neq \emptyset\}\,. \tag{2.15}$$

Accordingly, the function of *edge weights* has to be symmetrised with respect to i and j yielding

$$\widetilde{e} : \widetilde{E} \to \mathbb{R},\ \widetilde{e}(\,(B_i, B_j)\,) = \mu_{\mathrm{Leb}}(B_i) \cdot p_{ji} + \mu_{\mathrm{Leb}}(B_j) \cdot p_{ij}\,. \tag{2.16}$$

Note that by construction the total edge weight of the directed and the undirected transition graph is equal.

To measure the degree of invariance of a subset of vertices $S \subseteq V$, external and internal costs are defined. For two subsets of vertices $A, B \subseteq V$ the following notation is introduced

$$E_{A,B} = \frac{1}{2} \sum_{B_i \in A,\, B_j \in B} \left[\mu_{\mathrm{Leb}}(B_i) \cdot p_{ji} + \mu_{\mathrm{Leb}}(B_j) \cdot p_{ij}\right] \tag{2.17}$$

which simplifies for $A = B$ to $E_{A,A} = \sum_{B_i,\, B_j \in A} \mu_{\mathrm{Leb}}(B_i) \cdot p_{ji}$. Now, for $S \subseteq V$ the *internal costs* are defined as

$$C_{\mathrm{int}}(S) = \frac{E_{S,S}}{\mu_{\mathrm{Leb}}(S)}\,, \tag{2.18}$$

and the *external costs* as

$$C_{\mathrm{ext}}(S) = \frac{E_{S,\overline{S}}}{\mu_{\mathrm{Leb}}(S) \cdot \mu_{\mathrm{Leb}}(\,\overline{S}\,)} \tag{2.19}$$

where $\overline{S} = V \setminus S$ is the complement of S. Based on the previous definitions, for a partition of the vertices $\mathcal{P}(V) = \{P_1, \ldots, P_N\}$ the analogous quantities are the *internal costs*

$$C_{\mathrm{int}}(\mathcal{P}(V)) = \frac{1}{N} \sum_{i=1}^{N} C_{\mathrm{int}}(P_i) \tag{2.20}$$

and the *external costs*

$$C_{\mathrm{ext}}(\mathcal{P}(V)) = \frac{\sum_{1 \leq i < j \leq N} E_{P_i,\, P_j}}{\prod_{i=1}^{N} \mu_{\mathrm{Leb}}(P_i)}\,. \tag{2.21}$$

In [46] the internal and external costs are seen to be intuitively high and low, respectively, for almost invariant sets. However, it is stated that minimising the internal costs is not equivalent to maximising the external costs. Optimisation with respect to the first

criterion may lead to relatively small sets in the partition while optimisation with respect to the second criterion leads to more balanced partitions. Identifying a partition $\mathcal{P}(X) = \{P_1, \ldots, P_N\}$ with the corresponding one of the graph $\mathcal{P}(V) = \{P_1, \ldots, P_N\}$ yields $T_{\text{inv}}(\mathcal{P}(X)) = C_{\text{int}}(\mathcal{P}(V))$, and thus that maximising internal costs is equivalent to Problem 2.4. Minimising the external costs yields the third partitioning problem:

2.5 Problem (Discretised Partitioning into Almost Invariant Sets (II))
For a graph $G = (E, V)$ and fixed $N \in \mathbb{N}$ find a partition of vertices $\mathcal{P}(V) = \{P_1, \ldots, P_N\}$ with $\sum_{p \in P_i} v(p) > 0$ that minimises C_{ext}.

2.1.3 Complexity, Algorithmic Issues and Software

Most variants of the graph partitioning problems including the above mentioned are NP-complete [46]. Therefore, heuristics are employed to solve Problem 2.4 and Problem 2.5. An incomplete list of software tools for (balanced) graph partitioning may contain e. g. JOSTLE [211], METIS [91], and PARTY [166]. Throughout this thesis, graph partitioning will be performed using PARTY.

A common idea influencing the design of software packages is the *multilevel paradigm* which has been proven to be powerful (e. g. [141]). This paradigm consists of two steps: graph coarsening and local improvement. For the graph coarsening a heuristic approach called graph matching is employed in PARTY. A *graph matching* is a subset of edges such that each vertex is, at most, part of one edge. The coarsening procedure then reduces every two linked vertices of the matching to one *supervertex*. The hierarchical multi-step coarsening yields a clustering[(11)] on the coarsest level. It is followed by a stepwise projection onto the next finer level with a local optimisation of the partition of each level by standard methods like Kernighan-Lin [92] or the *Helpful-Set Method* [52].

The topic of partitioning with a variable number of partitions is even more challenging than the aforementioned partitioning problems. Some heuristics such as *congestion* [46] can be introduced but are not discussed further here.

2.2 Markov Decision Processes (MDPs)

Markov decision processes (MDPs) are the next step towards Markov games. They provide a model for situations where a stochastic discrete dynamical system evolves under the influence of *one* agent. In robot soccer this agent may also represent a whole team if it is fully cooperative. Typically, solving an MDP means to compute a global feedback control law to achieve some predefined goal in accord with the underlying dynamics. The textbooks [14, 168] provide a good introduction to the material, as does [202] where the notation of this section partly stems from.

Historical Remarks. MDPs were popularised by the books of Bellman and Howard [14, 84] but according to Puterman [168] the historical roots are located much earlier. Some of the basic concepts date back to problems of the calculus of variation to the 17th century but an explicit reference only points to the end of the 19th century: a paper of Cayley [30]. The beginning of the modern study can be dated to the 1940s where Wald [210] already presented the essence of the theory. A little later, important work was done on games

[(11)] The coarsening can be continued until the correct number of partitions is obtained, or until the coarsest level can be partitioned by standard partitioning methods.

[15, 186], stochastic inventory models [61], pursuit problems [86] and sequential statistical problems [5].

2.2.1 Basics and Problem Definitions

This subsection begins by defining the basic ingredients of an MDP:

2.6 Definition (Markov Decision Process)
A (discrete time, finite) Markov decision process $\mathcal{M} = (\mathcal{D}, \mathcal{S}, \mathcal{SA}, T, R)$ *is given by*

1.) decision epochs $\mathcal{D} = \mathbb{N}_0$,

2.) a (finite) state space $\mathcal{S}$,

3.) a (finite) state action space $\mathcal{SA} = \{(s, a) : s \in \mathcal{S}, a \in \mathcal{A}(s)\}$ *where* $\mathcal{A}(s)$ *is a (finite) set of available* actions *in state* s,

4.) a transition function $T : \mathcal{SA} \times \mathcal{S} \to [0, 1]$ *with* $T(s, a, s')$ *being the probability of reaching state* s' *if choosing action* a *in state* s,[(12)]

5.) a (deterministic) reward function $R : \mathcal{SA} \to \mathbb{R}$.

A standard assumption is that for every pair $(s, a) \in \mathcal{SA}$ holds that $\sum_{s' \in \mathcal{S}} T(s, a, s') = 1$ which means that no action can lead to a state outside of $\mathcal{S}$. The addition of (absorbing) extra states may help to ensure the above condition. For hierarchical problems as mentioned in Section 2.5 it can be necessary to differentiate between states within the same level of hierarchy (with a probability less than 1) and states outside of the given hierarchy level.

2.7 Remark (General MDP [168])
For a more general version of MDPs the following aspects may additionally be taken into account:

1.) The *decision epochs* can be either discrete or continuous and, in the first case, there is the possibility of a finite or an infinite set of time points; in the continuous case there are further possibilities when the decisions are to be made: continuously, at random time points, or at timepoints which itself can be decided by the decision maker.

2.) The *state space* can be continuous, discrete, or a mixture of both, and again finite or infinite in the discrete case.[(13)]

3.) The *action space* can have the same specifications as the state space and may be different for every state, additionally, instead of pure actions, so-called mixed action i. e. a *probability distribution* $\mathrm{PD}(B)$ on a (Borel) subset of actions can be performed during the Markov decision process. In this context, pure actions are degenerate mixed ones.

4.) The *reward function* may also be dependent upon the reached state. In the model of reward it is not important how the reward is acrued (continuously through a period, system state of subsequent decision epoch) but it or its expected value must be known

[(12)] An alternative for MDPs with deterministic state transitions is to define the modified transition function $\widetilde{T} : \mathcal{SA} \to \mathcal{S}$ with $T(s, a) = s'$ being the next state which is reached with probability 1.
[(13)] The exact condition is that the state space is a non-empty Borel subset of a complete, separable metric space. The same condition is needed for the action spaces. "Separable" means that there exists a countable dense subset, and a Borel set is an element of the Borel σ-algebra of the metric space.

before the next choice of action. If the reward depends on the subsequent state then it may be computed by

$$R(s,a) = \sum_{s' \in \mathcal{S}} T(s,a,s') R(s,a,s') \,. \tag{2.22}$$

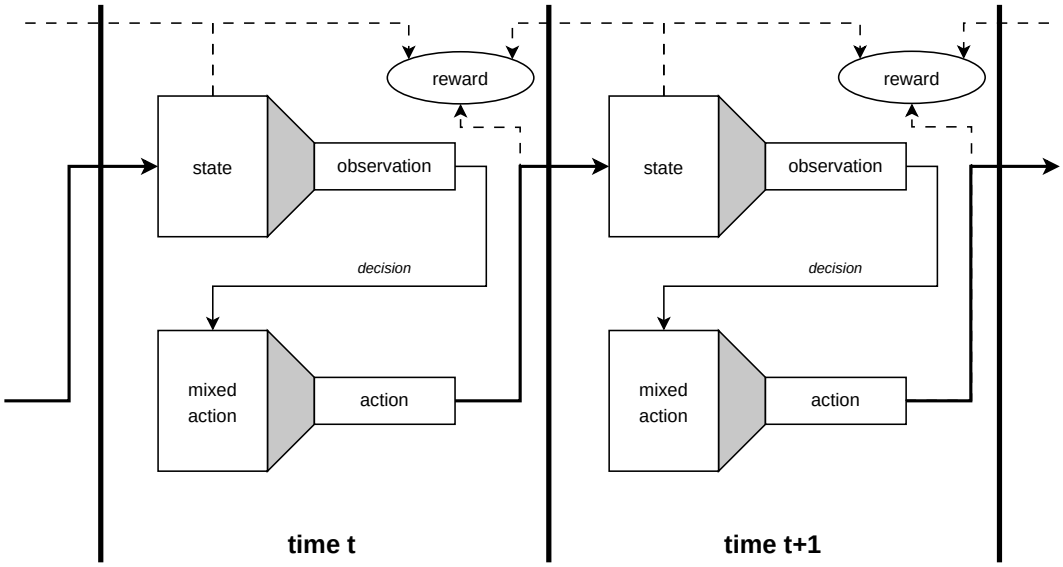


Figure 2.2: Scheme of a Markov decision process. The agent, being in a state at time t, bases its decision on a stochastic observation of the state to choose a mixed action. By random, this leads to a pure action and, again by random, to the next state. The agent receives (local) rewards depending on states and actions.

2.8 Definition (Markov Property)
A decision process is said to be Markovian *if for a sequence of state-actions $(s_t, a_t)_t$ with $s_t \in \mathcal{S}$ and actions $a_t \in \mathcal{A}(s_t)$, and a sequence of rewards $(r_t)_t$ with $r_t = R(s_t, a_t)$, the following holds:*

$$\begin{aligned} &\operatorname{Prob}\left\{ s_t = s, r_t = r \,|\, s_{t-1}, a_{t-1}, s_{t-2}, a_{t-2}, \ldots, s_0, a_0 \right\} \\ &\qquad = \operatorname{Prob}\left\{ s_t = s, r_t = r \,|\, s_{t-1}, a_{t-1} \right\} . \end{aligned} \tag{2.23}$$

An MDP possesses the Markov property because its transition function T and reward function R are just designed in this way. The Markov property means that the transitions and rewards are not dependent on history. A common artifice to include (a part of) the state action history in an MDP framework is to attach it directly to the state. It is obvious that this may augment the state space enormously depending on the maximal length of history information.

2.9 Example (Robot Soccer, 1)
Robot soccer will be the standard example throughout this thesis (for a more detailed description see Section 5.1) and serves to explain most of the theoretical concepts. If the strategy of the opponent team is fixed then only one team being represented by an abstract agent has to explicitly decide on its actions. This is exactly the situation of an MDP, and the ingredients of Definition 2.6 are as follows: the decision epoch $\mathcal{D} = \mathbb{N}_0$ is equidistant and infinite, the state space $\mathcal{S} \subseteq \mathbb{R}^n$ is the set which describes the possible coordinates

(position and maybe velocity) of all robots and the ball, the action set $\mathcal{A}(s)$ consists of movements and kicks, the (probabilistic) transition function T describes the evolution of the game for each state and each action, and the reward R is simply positive ($= +1$) for scoring a goal and negative ($= -1$) for letting the opponent team score a goal.

Return Models

After the definition of the dynamics and reward of an MDP the goal of an MDP is to be stated: to maximise some kind of long-term reward called the *return* $\mathcal{R}$.[14] Given a stochastic[15] sequence of rewards $(r_t)_t$ there are, however, different criterions of optimality. In [88, 202], the following variants of measuring the optimality by returns are considered: Wide spread due to convergence properties is the *discounted infinite horizon return*

$$\mathcal{R}_{\text{disc}} = \mathrm{E}\left\{\sum_{t=0}^{\infty} \gamma^t r_t\right\} \tag{2.24}$$

with *discount rate* $\gamma \in (0, 1)$, while the numerical approximation of the infinite horizon return often is performed by the corresponding *finite horizon return*

$$\mathcal{R}_{\text{disc}}^{N} = \mathrm{E}\left\{\sum_{t=0}^{N} \gamma^t r_t\right\} \tag{2.25}$$

where additionally $\gamma = 1$ is allowed.[16] $\mathrm{E}\{\}$ here means the *expectation value*. Completely different and more difficult to analyse are the *average infinite horizon return* model

$$\mathcal{R}_{\text{aver}} = \lim_{N\to\infty} \mathrm{E}\left\{\frac{1}{N+1}\sum_{t=0}^{N} r_t\right\}, \tag{2.26}$$

or bias-optimal models which both do not need a discount factor. An empirical comparison of a discounted to an average reward based return model can be found in [125].

At this point it should be remarked that all of the following concepts which deal with optimality depend strongly on the optimisation criterion defined by the return model. Therefore, if not stated differently, the standard is to assume the discounted infinite horizon return, i. e. $\mathcal{R} = \mathcal{R}_{\text{disc}}$.

2.10 Example (Robot Soccer, 2)
Using the robot soccer example again for illustration, a discounted reward means that scoring a goal faster is ranked higher than scoring it later. The finite horizon version seems to be adequate if the duration of the soccer match and of the performance of actions can be exactly foreseen – which is typically not true. Finally, in the average reward case the ranking of scoring is completely independent of the exact time points. Only the average amount of goals per time interval is important.
It is also possible to design arbitrarily return profiles with weights $w_t \neq \gamma^t$ as long as convergence of the series can be guaranteed. This may be useful to enforce some strategic

[14] Sometimes, the MDP as defined above is called Markov decision *process* and the MDP plus return model is called Markov decision *problem*. In this thesis, the term Markov decision process includes also the return model.
[15] The rewards are either stochastic themself or by the stochasticity of the transition function, or both.
[16] In the discounted infinite horizon return model one can also consider $\lim_{\gamma\to 1}$ if existent.

behaviour for which a desired time scale is specified (*soft constraint*). However, for a non-constant weight the Markov property of the value function is typically not fulfilled. Although not considered before, employing such a method may give additional tactical options for the soccer game.

Policies, Value Functions, and Optimality

A definition of the concepts of optimality remains: policies, value functions, and Bellman's principle of optimality.

2.11 Definition (Policy, Decision Rule)
A (stationary Markovian) policy $\pi : \mathcal{SA} \to [0,1]$ *is a decision rule which for each* $s \in \mathcal{S}$ *specifies a probability distribution* $\mathrm{PD}(\mathcal{A}(s))$*, i. e.* $\forall s \in \mathcal{S} : \sum_{a \in \mathcal{A}(s)} \pi(s,a) = 1$*.*

According to the definition a policy π specifies the probability $\pi(s,a)$ of performing action a in state s. If $\pi(\mathcal{SA}) = \{0,1\}$, then the policy is called *deterministic*. Given an initial state $s_0 \in \mathcal{S}$ and a policy π, one can compute an associated random trajectory $\mathcal{O}(s_0) = (s_t, a_t)_{t \in \mathbb{N}_0}$ with rewards $r_t = R(s_t, a_t)$ and the corresponding return. Then, the state value function $V^\pi : \mathcal{S} \to \mathbb{R}$ is the expectation of the return under policy π when starting in state s. Similarly, the *state-action value function* $Q^\pi(s,a)$ is the expectation of the return under policy π when starting in state-action (s,a). A formal definition follows:

2.12 Definition (Value Functions of an MDP)
Given an MDP $\mathcal{M} = (\mathcal{D}, \mathcal{S}, \mathcal{SA}, T, R)$ *with* $\mathcal{D} = \mathbb{N}_0$ *and a policy* π*, the state value function* $V^\pi : \mathcal{S} \to \mathbb{R}$ *under this policy is defined by*

$$V^\pi(s) = \mathrm{E}_\pi \left\{ \mathcal{R} \,|\, s_0 = s \right\} \tag{2.27}$$

where the notation $\mathrm{E}_\pi \{X\} = \mathrm{E}\left\{X \;\middle|\; \forall t : \mathrm{Prob}\left\{s_{t+1} = s' \,|\, s_t = s, a_t = a\right\} = T(s,a,s')\right.$ and $\left.\mathrm{Prob}\left\{a_t = a \,|\, s_t = s\right\} = \pi(s,a)\right\}$ *is employed. The corresponding state action value function* $Q^\pi : \mathcal{SA} \to \mathbb{R}$ *under this policy is defined by*

$$Q^\pi(s,a) = \mathrm{E}_\pi \left\{ \mathcal{R} \,|\, s_0 = s, a_0 = a \right\}. \tag{2.28}$$

A policy π^* *is called optimal if for all policies* π *holds* $V^{\pi^*} \geq V^\pi$*.*[17]

The above definition is not well-suited to reveal a method for algorithmically computing value functions. Most of the solution algorithms utilise a recursivity property known as Bellman's principle of optimality.

2.13 Theorem (Optimality Principle [168])
A (state-) value function $V \in \mathbb{R}^{|\mathcal{S}|}$ *of an MDP is optimal iff it is the unique solution to the Bellman equation:*

$$V(s) = \max_{\pi_s \in \mathrm{PD}(\mathcal{A}(s))} \sum_{a \in \mathcal{A}(s)} \pi_s(a) \cdot Q(s,a) \tag{2.29}$$

where

$$Q(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') \cdot V(s'). \tag{2.30}$$

Here, $\mathrm{PD}(X)$ *is the set of probability distributions on the set* X *and* π_s *is the restriction of* π *to state* s *meaning* $\pi_s(a) = \pi(s,a)$*.*

[17] The optimality condition is equivalent to the following one: $Q^{\pi^*} \geq Q^\pi$ for all policies π as can be seen by plugging Q^*, Q^π into Equation 2.29 and, reversely, V^*, V^π into Equation 2.30.

The result of plugging Equation 2.30 into Equation 2.29 can be abbreviated by the *Bellman operator* $\mathcal{B}_{\mathrm{MDP}}$ which shortens the notation of the Bellman equation to $V = \mathcal{B}_{\mathrm{MDP}}V$. The Bellman equation is a nonlinear fixed point equation for the optimal value function $V^* = V^{\pi^*}$ and the operator is a contraction in $\| \|_\infty$ with rate γ [19]. Note that every MDP has a deterministic optimal policy [168] and it would thus suffice to take the maximum in Equation 2.29 only over *pure actions* $a \in \mathcal{A}(s)$. However, in the case of 2P-ZS-MGs in Section 2.4 a need for the more general formulation will arise. In order to stress the formal similarities the notation above deviates from standard.

2.14 Remark (History Dependent Strategies, Non-Stationarity [168])
As stated, policies can be deterministic or stochastic. One concept which is not covered by the policy of Definition 2.11 are history dependent strategies. A t-time history h_t is recursively defined by $h_0 = s_0$ and $h_i = (h_{i-1}, a_{i-1}, s_i)$. A second concept which is also not covered is the possible non-stationarity of policies meaning that it also explicitly depends on time. Both concepts are irrelevant to find the optimal solution of an MDP but can be of practical interest if the Markov condition is not (exactly) fulfilled.

2.2.2 Numerical Methods

Two important classes of approaches are *dynamic programming* (DP) and *reinforcement learning* (RL) methods. While for the first class of solution methods it is assumed that the stochastic model is known in advance, in the second class this assumption is dropped and the consequences of the world model are directly (model-based approaches) or indirectly (model free approaches) approximated.

Dynamic Programming Methods

2.15 Definition (Value Iteration (MDP) [168])
The following algorithm is called value iteration*: select $\varepsilon > 0$, choose an arbitrary initial guess $V_0 \in \mathbb{R}^{|\mathcal{S}|}$ for the (state-) value function, and determine iteratively $V_k = \mathcal{B}_{\mathrm{MDP}}V_{k-1}$ for $k = 1, 2, \ldots$ until $\|V_{k+1} - V_k\|_\infty \leq \frac{\varepsilon}{2} \cdot \frac{1-\gamma}{\gamma}$.*

The stopping criterion can also be based on a span semi-norm [168] which may improve the contraction rate if the transition matrix is non-sparse. Value iteration converges to V^*, and provides an $\frac{\varepsilon}{2}$-approximation for the value function estimate, i. e. $\|V_N - V^*\|_\infty \leq \frac{\varepsilon}{2}$, and an ε-optimal stationary policy by

$$\pi^*_\varepsilon(s) = \Pi^*(V_N) = \arg\max_{a \in \mathcal{A}(s)} \big(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s') \cdot V_{N-1}(s')\big) \tag{2.31}$$

where N is the number of iterates and $\arg\max_{a\in\mathcal{A}(s)} \in \mathrm{PD}(\mathcal{A}(s))$ is considered to be a probability distribution of a pure action [168]. In the case of several equally good actions the "arg max" has to return a mixed action with equal probabilities to keep uniqueness. Note that in terms of the state-action value function Equation 2.31 can also be written $\pi^*_\varepsilon(s) = \Pi^*(Q_{N-1}) = \arg\max_{a\in\mathcal{A}(s)} Q_{N-1}(s,a)$.[18]

Furthermore, the convergence is linear at rate γ, i. e. $\|V_{k+1} - V^*\| \leq \gamma \cdot \|V_k - V^*\|$, and that convergence for Gauss-Seidel value iteration, which uses $V_{k+1}(s)$ instead of $V_k(s)$ as

[18] Analogue to later proofs the iterative procedure of applying the two parts of the Bellman equation (Equations 2.29 and 2.30) is labelled in the order: V_0, Q_0, V_1, Q_1, ...

soon as available, is linear at least with rate γ [168]. The analysis of the algorithms is based on the fact that for each iteration k there exists a policy π_k which corresponds to the maximum selection rule. With respect to these policies π_k the analysis is done in a way similar to iterative solvers for linear equations because the evaluation of a fixed policy is solving a linear equation with the matrix depending on the policy (Appendix B).

For the sake of completeness and because of its superlinear, sometimes quadratic convergence the policy iteration algorithm is presented. For MDPs it is further guaranteed to find the optimal policy in finitely many steps but this relies on the existence of a deterministic optimal policy and is not true for 2P-ZS-MGs.

2.16 Definition (Policy Iteration [168])
Select an arbitrary policy π_0, and repeat the following until $\pi_{k+1} = \pi_k$: compute the value function $V_k = V^{\pi_k}$ of policy π_k (policy evaluation) and choose the policy $\pi_{k+1} = \Pi^*(V_k)$ as in Equation 2.31 (policy improvement).

Reinforcement Learning Methods

Reinforcement learning methods [88] belong to the class of stochastic approximation algorithms and are employed for numerical comparisons. From a practical point of view they may help to adopt optimal policies of a DP model to a real world problem. A key difference of RL is that the agent directly executes actions according to a policy and, based on the outcomes, estimates the value function and in some cases a model of the underlying MDP. One of the most common algorithms is Q-learning first proposed by [212]:

2.17 Definition (Q-Learning [202])
Start with state-action value function Q_0, and let the agent be in an initial state s_0. Repeat iteratively: Being at step k in state s_k, choose an action a_k according to an arbitrary policy π, observe the next state s_{k+1} and set

$$Q_{k+1}(s,a) = \partial_{(s,a),(s_k,a_k)} \cdot \Big[(1-\alpha_k)Q_k(s_k,a_k) + \alpha_k(R(s_k,a_k) + \gamma \cdot \max_{a_{k+1}\in\mathcal{A}(s_{k+1})} Q_k(s_{k+1},a_{k+1}))\Big]. \quad (2.32)$$

where $\partial_{i,j}$ is the Kronecker symbol being 1 if $i = j$ and 0 else, and $\alpha_k = \alpha_k(s,a)$.

The Q-learning rule is guaranteed to converge with probability 1 (for any initial state s_0 and for any Q_0) if for the sequence of learning rates holds for every $(s,a) \in \mathcal{SA}$ [19]:

$$\sum_{k=0}^{\infty} \alpha_k(s,a) = \infty, \ \sum_{k=0}^{\infty} \alpha_k^2(s,a) < \infty\,. \quad (2.33)$$

The first condition includes the necessity for infinitely updating every state-action pair while the second one establishes convergence. Because the convergence to the optimal Q-value function occurs for *every* policy π of the agent as long as Equation 2.33 is fulfilled, the method is called an *off-policy method*. In practice, *exploration strategies* are included to guarantee Equation 2.33 while *exploitation*, i. e. performing an optimal action, ensure that the most promising parts of the state (-action) space are updated faster and more often. A very common policy is the *ε-greedy* which chooses with probability $\varepsilon > 0$ a random action while greediness means to choose an optimal action $a_k = \Pi^*(Q_k)$ with probability $1 - \varepsilon$.

Other possibilities are to introduce an exploration bonus, curiosity driven exploration [183], Boltzman exploration or interval based techniques [38].

For the sake of completeness a common *on-policy method* is to be introduced. It is called *SARSA*, which stems from the update formula (state, action, reward, state, action), and is similar to Q-learning but with the slightly modified update rule:

$$Q_{k+1}(s,a) = \partial_{(s,a),(s_k,a_k)} \cdot \Big[(1-\alpha_k)Q_k(s_k,a_k) + \alpha_k(R(s_k,a_k) + \gamma \cdot Q_k(s_{k+1},a_{k+1}))\Big]. \quad (2.34)$$

The difference is now that if the agent would execute policy π, the SARSA algorithm would converge to Q^π instead of Q^*. However, if the policy is not kept constant but instead the agent follows an ε-greedy policy with respect to the Q-values and ε is decayed over time then it can be hoped [202] that SARSA converges to Q^*. Compared to DP methods SARSA perhaps can be interpreted as an asynchronous version of a policy iteration algorithm.

[116, 203] give general conditions under which asynchronous RL algorithms converge. They reduce the effort to prove the convergence of synchronous update rules. One special case is Q-learning for MDPs.

2.2.3 Complexity, Algorithmic Issues and Software

The numerical effort of solving MDPs is different for different algorithms [88]: Value iteration needs in the worst-case per iteration $O(|\mathcal{SA}| \cdot |\mathcal{S}|)$ multiplications and evaluations of T or, if the transition function is sparse ($T(s,a,s') \neq 0$ only for constantly many s'), then value iteration only needs $O(|\mathcal{SA}|)$ multiplications and evaluations of T. However, the number of iterations to achieve some prescribed ε-optimality can grow dramatically if the discount factor γ approaches 1 (see stopping criterion of Definition 2.15). In practice, policy iteration can converge faster although the per iteration complexity is $O(|S|^3 + |\mathcal{SA}| \cdot |S|)$ (policy evaluation + improvement) and there are no general theoretical worst-case results [115]. For special cases like deterministic MDPs Madani gives proofs that policy iteration is in P (polynomial) [123] and so is value iteration.

Linear Programming [184] is also a method to solve MDPs with the advantage that very efficient commercial packages can be used. Theoretically, this is the only known method with polynomial time although this need not indicate that it is the most efficient in practice. Other standard ideas to speed up numerical techniques can be used e. g. *multi-grid methods* as in [175] or *state aggregation* which conglomerates several states to a single meta-state [18].

The Q-learning method presented above belongs to the model free approaches as well as adaptive heuristic critic [12]. Under some conditions model free methods are applicable in the average return case [185]. Model-based approaches which were not introduced here include certainty equivalence [95], Dyna-Q [199, 200, 213], *Queue Dyna* [163], and *Prioritised Sweeping* [144, 44]. Special model-based methods for finding short paths to a goal state are real-time dynamic programming (RTDP) [10] and plexus planning [43].

2.3 Matrix Games

In general, game theory is a scientific field which deals with the question: How should two or more agents, being in an intertwined situation, decide to their own best?[19] In a less selfish formulation the question may also be stated as "how to model and solve conflicts". Similar to MDPs, there are different possibilities concerning the rules and settings of the *game* situations. Again, one of the most distinguishing criteria is the question of whether the game is a *differential game*, which is time-continuous, or a (time-) *discrete game*. A second criterion is whether the game is deterministic or stochastic. For a more detailed overview one may consult the textbooks [13, 66, 86, 107, 158, 159] and the article [135].

Matrix games are a very special and restricted class of games but nevertheless helpful to begin with. They belong to the class of two-player zero-sum games whereas only one time step is under consideration (*single-stage game*), or whereas the situation does not change with time (repeated matrix games). From a theoretical point of view repeated matrix games are different from normal ones if the allowed strategies are deterministic but may be changed over time. In such a case non-Markovian strategies like "tit-for-tat", which depend on the action history, are sometimes reasonable and offer different options than Markovian stochastic strategies (*mixed actions*) that are independent from time. In the sequel, matrix games are a basic element for the solution of the more general class of Markov games. This is the reason why they are introduced and why a numerical method for solving this class of games is presented.

Generally, two-player zero-sum games can be interpreted as games between two rivals: one agent wants to avoid what the other tries to achieve and vice versa. Sometimes, such a situation is called (completely) *competitive* because there is no potential for a compromise. Typically, board games (backgammon, chess) or sport games (soccer, tennis) are of this type because one agent or team of agents wins if and only if the other one loses.

Historical Remarks. [13] dates the theory of zero-sum games back to the 1920s when Borel did some work which was translated into English much later [20]. Borel developed some concepts of strategies but conjectured that the minimax theorem (Theorem 2.23) was false. Von Neumann proved the opposite [154] – the first proof of the minimax theorem was not as elementary as e. g. in [13] – and built the foundations of game theory which are summarised in his famous book with Morgenstern [155]. Other early references include [120, 136] and the work of Nash.

2.3.1 Basics and Problem Definitions

Matrix games are finite two-player zero-sum games in *normal form* whereas the *extensive form* is represented by a tree structure [13]. This tree structure allows a more detailed picture e. g. about the order of play and the information of the agents at each decision epoch. However, for the Markov games considered later the standard assumption will be that at every time step the actions of all agents take place simultaneously or, equivalently, that one agent does not know the action of the other agent before the end of a decision epoch. Thus, the representation of each time step by a matrix game is adequate. To distinguish between zero-sum and general-sum games with two agents the latter ones are called *bimatrix games* [66] to indicate that for each agent P_i, $i = 1, 2$ a *reward matrix* or *payoff matrix* R_i has to be specified. Below, the definitions of a bimatrix game and a (zero-

[19] In game theory, agents are typically called *players*.

sum) matrix game are given and the interchangeability theorem is stated. Basic concepts like the value of a game, the minimax theorem (Theorem 2.23), and the dominance of actions are introduced.

Bimatrix and Matrix Games

The definitions start with bimatrix games which are two-player general-sum games in normal form and their zero-sum variant called matrix games. It is intended to keep analogies to MDPs (Definition 2.6) and, hence, slightly deviate from standard notation. Many of the following statements and definitions stem from or are inspired by [66].

2.18 Definition (Bimatrix Game)
A bimatrix game Γ *is defined by*

1.) a trivial decision epoch $\mathcal{D} = \{t_0\}$,

2.) a trivial state space $\mathcal{S} = \{s_0\}$,

3.) a finite state action space $\mathcal{SAO} = \{(s, a, o) : s \in \mathcal{S}, a \in \mathcal{A}(s), o \in \mathcal{O}(s)\}$ *where* $\mathcal{A}(s), \mathcal{O}(s)$ *are the (finite) sets of available actions of the first and the second agent,* P_1 *and* P_2*, respectively,*

4.) a trivial transition function $T : \mathcal{SAO} \times \mathcal{S} \to [0, 1]$ *with* $T(s_0, a, o, s_0) = 1$,

5.) a (deterministic) reward function *for* each *agent* P_i *denoted by* $R_i : \mathcal{SAO} \to \mathbb{R}$.

A *repeated bimatrix game* only differs in the decision epoch $\mathcal{D} = \mathbb{N}_0$ from a bimatrix game. The name bimatrix game stems from the fact that the only non-trivial elements are the finitely many actions of both agents and the two reward functions R_i that can be represented by two matrices. The convention will be to write the matrices in such a way that actions $\mathcal{A} = \{a_1, \dots, a_m\}$ of agent P_1 correspond to rows and actions $\mathcal{O} = \{o_1, \dots, o_n\}$ of agent P_2 correspond to columns of the matrices R_i as the following scheme indicates:

$$\begin{array}{c} \\ a_1 \\ a_2 \\ \vdots \\ a_m \end{array} \begin{array}{c} \begin{array}{cccc} o_1 & o_2 & \dots & o_n \end{array} \\ \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \dots & r_{mn} \end{pmatrix} \end{array} .$$

2.19 Definition (Matrix Game)
A matrix game *is defined to be a bimatrix game* Γ *with the additional condition that the sum of the reward matrices of the two agents is zero:*

$$R_1 + R_2 = 0\,. \tag{2.35}$$

A matrix game belongs to the class of two-player zero-sum games in normal form. Analogously to bimatrix games, a *repeated matrix game* only differs in the decision epoch $\mathcal{D} = \mathbb{N}_0$. As for MDPs, each agent P_1, P_2 has to find an optimal policy π_1^*, π_2^* which maximises its return $\mathcal{R}_1, \mathcal{R}_2$, respectively. In the case of non-repeated (bi)matrix games this is simply the selection of a mixed action[(20)], whereas the return $\mathcal{R}_i$ of agent P_i reduces to the expectation over the single-stage return (only first summand in the sum of e. g. Equation 2.24).

[(20)] The set of mixed actions for agent P_1 is the $(m-1)$-dimensional simplex of probability distributions $\mathrm{PD}(\mathcal{A}(s_0)) = \{x \in \mathbb{R}^m : x = (x_j)_j \text{ with } x \geq 0,\ \sum_{j=1}^m x_j = 1\}$, and for agent P_2 it is the analogue $(n-1)$-dimensional simplex.

If the policies of the two agents are represented by column vectors ($\pi_1 \in \mathbb{R}^{m,1},\ \pi_2 \in \mathbb{R}^{n,1}$) and the reward functions R_i by matrices as depicted above, then the expectation of the single-stage return $\mathcal{R}_i$ for agent P_i can be expressed simply by matrix vector multiplications:

$$\mathrm{E}_{\pi_1,\pi_2}\{\mathcal{R}_i\} = \pi_1^T R_i\, \pi_2 = \sum_{k,l} (\pi_1)_k\, (R_i)_{kl}\, (\pi_2)_l \tag{2.36}$$

with A^T denoting the transpose of a matrix A.

Before the interchangeability theorem is stated the general definition of a Nash equilibrium has to be introduced. More details under which conditions for the policy spaces Nash equilibria exist can be found in [22]. For defining a *Nash equilibrium* very briefly, the concept of best response is employed which is also applicable to the general case of k agents. A policy π_i^* of agent P_i is an element of the set of *best response* or *best reply* $BR_i(\pi_{-i})$ to a joint policy π_{-i} of all other agents if for all policies π_i holds that

$$\mathrm{E}_{(\pi_i^*,\pi_{-i})}\{\mathcal{R}_i\} \geq \mathrm{E}_{(\pi_i,\pi_{-i})}\{\mathcal{R}_i\} \tag{2.37}$$

with (π_i^*, π_{-i}) being appropriately ordered. Then, a Nash equilibrium is a tuple of policies $\pi^* = (\pi_1^*, \ldots, \pi_k^*)$ of the k agents with

$$\forall i:\ \pi_i^* \in BR_i(\pi_{-i}^*) \tag{2.38}$$

which means that no agent has an incentive to deviate unilaterally from its policy. A difficulty in determining Nash equilibria, which is in some cases NP-hard [34, 32], arises because the set of best response depends on the joint policy π_{-i} of all other agents. An exception is the computation of Nash equilibria in (two-player zero-sum) matrix games which is polynomial due to the solvability by linear programming [34]. A more precise statement of the complexity of finding Nash equilibria in n-player games with $n \geq 4$ gives [40]. More details on the complexity of linear programming can be found below in the subsection concerning numerical methods.

An important fact of arbitrary *two*-person zero-sum games which is not restricted to matrix games but is not true for general-sum games[(21)] is about the interchangeability of equilibrium pairs of strategies where equilibrium pair means Nash equilibrium. The theorem below shows that all Nash equilibria are equally good and, hence, that a Nash equilibrium is already a pair of globally optimal policies for both players. A pair of policies may also be called a *total policy* $\pi = (\pi_1, \pi_2)$.

2.20 Theorem (Interchangeability and Equal Payoff of Equilibrium Strategies [159]) *In a two-person zero-sum game Γ let (π_1, π_2) and $(\widetilde{\pi}_1, \widetilde{\pi}_2)$ be two pairs of equilibrium strategies. Then $(\pi_1, \widetilde{\pi}_2)$ and $(\widetilde{\pi}_1, \pi_2)$ are also equilibrium pairs, and for $i = 1, 2$ holds:*

$$\mathrm{E}_{\pi_1,\pi_2}\{\mathcal{R}_i\} = \mathrm{E}_{\widetilde{\pi}_1,\widetilde{\pi}_2}\{\mathcal{R}_i\} = \mathrm{E}_{\pi_1,\widetilde{\pi}_2}\{\mathcal{R}_i\} = \mathrm{E}_{\widetilde{\pi}_1,\pi_2}\{\mathcal{R}_i\}\ . \tag{2.39}$$

The proof only uses the zero-sum property and the general equilibrium inequalities which are valid for any strategy $\widehat{\pi}_1, \widehat{\pi}_2$, respectively: $\mathrm{E}_{\pi_1,\pi_2}\{\mathcal{R}_1\} \geq \mathrm{E}_{\widehat{\pi}_1,\pi_2}\{\mathcal{R}_1\}$ and $\mathrm{E}_{\pi_1,\pi_2}\{\mathcal{R}_1\} \leq \mathrm{E}_{\pi_1,\widehat{\pi}_2}\{\mathcal{R}_1\}$ to show the equality and that the new strategies are equilibria.

[(21)] Consider e. g. the bimatrix game with $R_1 = R_2 = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}$ and $a > b > 0$ which is sometimes called *coordination game* and has two Nash equilibria with expected return of a and b being equal for both agents.

2.21 Definition (Value of a Matrix Game [66])
A two-person zero-sum game Γ *is said to have a value* V^* *if and only if*

$$\sup_{\pi_1}\inf_{\pi_2}\ \mathrm{E}_{\pi_1,\pi_2}\{\mathcal{R}_1\} = \inf_{\pi_2}\sup_{\pi_1}\ \mathrm{E}_{\pi_1,\pi_2}\{\mathcal{R}_1\}\,(:= V^*)\,. \tag{2.40}$$

This means, agent P_1 can assure itself a return $\mathcal{R}_1 = V^*$ when acting optimally independent of what the other agent does. Vice versa, agent P_2 can assure itself a return $\mathcal{R}_2 = -\mathcal{R}_1 = -V^*$.[(22)] The left-hand side of Equation 2.40 is called the *lower value* of the game and the right-hand side the *upper value*. The upper and lower value represent *security levels* of the two agents.

For a matrix game $\Gamma = [M]$ with value $V^*(M)$, the policies $\pi_{1,\varepsilon}, \pi_{2,\varepsilon}$ are called to be ε-optimal ($\varepsilon \geq 0$) for agents P_1 and P_2 if

$$\inf_{\pi_2}\ \mathrm{E}_{\pi_{1,\varepsilon},\pi_2}\{\mathcal{R}_1\} \geq V^* - \varepsilon \text{ and } \sup_{\pi_1}\ \mathrm{E}_{\pi_1,\pi_{2,\varepsilon}}\{\mathcal{R}_1\} \leq V^* + \varepsilon\,, \tag{2.41}$$

respectively. 0-optimal strategies are called *optimal*. The *set of optimal strategies* for player P_k of a matrix game $\Gamma = [M]$ with matrix M is denoted by $\mathcal{O}^k(M)$ and the ε-optimal set by $\mathcal{O}^k_\varepsilon(M)$.

The following proposition can be obtained elementarily:

2.22 Proposition (Optimality of Saddle Points [66])
Be $\Gamma = [M]$ *a matrix game. The following holds:*

1. $\sup_{\pi_1}\inf_{\pi_2}\mathrm{E}_{\pi_1,\pi_2}\{\mathcal{R}_1\} \leq \inf_{\pi_2}\sup_{\pi_1}\mathrm{E}_{\pi_1,\pi_2}\{\mathcal{R}_1\}\,.$
2. *(Optimality of Saddle Points) When there exist policies* π_1^*, π_2^* *such that for all* π_1, π_2*:* $\mathrm{E}_{\pi_1,\pi_2^*}\{\mathcal{R}_1\} \leq \mathrm{E}_{\pi_1^*,\pi_2^*}\{\mathcal{R}_1\} \leq \mathrm{E}_{\pi_1^*,\pi_2}\{\mathcal{R}_1\}$*, then the value of the game exists and* π_1^*, π_2^* *are optimal.*

An essential theorem for matrix games is the following one which was beneath others shown by J. von Neumann ([154], comment in [209]):

2.23 Theorem (Minimax Theorem for Matrix Games [66])
For every matrix $M \in \mathbb{R}^{m,n}$ *the corresponding matrix game* $\Gamma = [M]$ *has a value* $V^* = V^*(M)$ *and both agents* P_1, P_2 *possess optimal strategies* $\pi_1^* \in \mathbb{R}^m, \pi_2^* \in \mathbb{R}^n$*.*

A relatively simple *direct* proof of the minimax theorem can be found in [13]. More elegantly, the theorem is obtained by duality in linear programming [194].[(23)]

A generalisation of convergence properties from MDPs to 2P-ZS-MGs [203] needs a non-expansion property of determining the value of a game in terms of the matrix entries. In view of this, the following matrix distance, which does not coincide with the $\|\ \|_\infty$-norm for matrices, is introduced.

2.24 Definition (Distance of Game Matrices)
For two matrices $M_1, M_2 \in \mathbb{R}^{m,n}$ *the* game matrix distance d_Γ *is defined by*

$$d_\Gamma(M_1, M_2) = \max_{i,\,j}|(M_1 - M_2)_{ij}|\,. \tag{2.42}$$

[(22)] The notation deviates from standard as far as throughout this thesis every agent maximises its own return, which is in the two-player zero-sum case equivalent to minimise the return of the other agent.
[(23)] The duality theorem is stated as Theorem 2.30.

2.25 Proposition (Properties of Matrix Games [66])
Let $M, M_1, M_2 \in \mathbb{R}^{m,n}$ be matrices and $J \in \mathbb{R}^{m,n}$ denote the matrix with $J_{ij} = 1$ for all i, j.

1.) (Addition of Constants) For any $c \in \mathbb{R}$: $V^*(M + cJ) = V^*(M) + c$, and the optimal strategy sets $\mathcal{O}^1, \mathcal{O}^2$ for both players are the same for the matrix games $[M]$ and $[M + cJ]$. Thus, the assumptions $M_{ij} > 0$ and $V^*(M) > 0$ are not restrictions.

2.) (Monotonicity) If $(M_1)_{ij} \leq (M_2)_{ij}$ for all i, j then $V^*(M_1) \leq V^*(M_2)$.

3.) (Non-Expansion, Continuity of Value) It holds $|V^*(M_1) - V^*(M_2)| \leq d_\Gamma(M_1, M_2)$.

A reduction of matrix games can sometimes be achieved by eliminating dominated actions. While this is theoretically interesting and may lead to non-trivial reductions if applied recursively, the numerical effort of comparing each row to every other and likewise for the columns seems to only be appropriate if a direct numerical solution of a matrix game fails.

2.26 Definition (Dominance of Actions [13])
Let $\Gamma = [M]$ be a matrix game. The i-th action of player P_1 (row of matrix M) is said to dominate *the j-th action (row) if $e_i^T M \geq e_j^T M$ (i. e. for all k: $m_{ik} \geq m_{jk}$) and in one component the inequality is strict. Similarly, the i-th action of player P_2 (column of matrix M) is said to dominate the j-th action (column) if $Me_i \leq Me_j$ and in one component the inequality is strict. The actions are called* strictly dominating *if for all components inequality is strict.*

2.27 Theorem (Elimination of Dominant Actions [159])
Let $\Gamma = [M]$ be a matrix game, and assume that rows $i_1, \ldots, i_k$ of M are dominated. Then agent P_1 has an optimal policy π_1 with $(\pi_1)_{i_1} = \cdots = (\pi_1)_{i_k} = 0$. Moreover, any optimal policy for the game obtained by removing the dominated rows will also be an optimal policy for the original game.

The analogous statement for agent P_2 results from applying the theorem to $-M^T$. [13] gives an example that deleting non-strictly dominating actions can reduce the number of optimal strategies but as long as the aim is to compute the value of a game and only *one* optimal strategy this raises no problem.

2.28 Example (Robot Soccer, 3)
A well-known matrix game is *matching pennies* which is given by the matrix

$$M = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{2.43}$$

In a robot soccer environment the same type of matrix game may arise if one simplifies a situation where a robot tries to dribble the ball around one opponent robot. The decision of the dribbling agent is to go left (a_1) or right (a_2), while the opponent agent has to decide to block left (o_1) or right (o_2). If both choose the same direction then the dribbler loses the ball, otherwise the move is successful.
It is also possible to consider the complete tactics of a soccer game as an action of a matrix game. More precisely, the actions are to choose the tactics before the game starts and the reward matrix is related to the expected scoring at the end of the game. Obviously, this concept is not suitable to initially construct different tactics and the tactics of particular game situations are not alterable. Furthermore, obtaining the expectations of all strategy combinations requires a lot of games to be played.

2.3.2 Numerical Methods

Bimatrix games can be solved by the *Lemke-Howson algorithm*, see e. g. the excellent survey of (bi)linear methods for solving bimatrix games in [194] or the original work [106], or by the Mangasarian-Stone algorithm (Equation 2.46).[(24)] Accordingly, it is possible to numerically solve matrix games by linear programming. An overview of linear programming gives [208] as well as the textbook [39] by Dantzig who was the pioneer of the simplex method.

Linear Programming (LP) and Matrix Games

For determining the optimal value function V^* of a Markov game it is necessary to calculate the value of matrix games. Numerically, the solution of matrix games is addressed by LP, which motivates the following description. It includes a definition of linear programs, the duality theorem, the solution of matrix games[(25)], and concludes with the Mangasarian-Stone algorithm for the solution of bimatrix games.

2.29 Definition (Primal and Dual Linear Program)
A linear program is determined by a triplet (M, b, c) where $M \in \mathbb{R}^{m,n}, b \in \mathbb{R}^m, c \in \mathbb{R}^n$. This triplet defines two related optimisation problems:

$$\begin{array}{lll} \textit{Primal} & & \textit{Dual} \\ \textit{maximise} & c^T x & \textit{minimise} \quad b^T y \\ \textit{subject to} & Mx \leq b & \textit{subject to} \quad M^T y \geq c \\ & x \geq 0 & \qquad\qquad y \geq 0 \end{array} \tag{2.44}$$

The sets $S_p = \{x \in \mathbb{R}^n \,|\, Mx \leq b, x \geq 0\}$ and $S_d = \{y \in \mathbb{R}^m \,|\, M^T y \geq c, y \geq 0\}$ are called *feasible sets* for the primal and dual program, respectively. Elements of these sets are *feasible solutions*, and the primal and dual linear program is called feasible if the corresponding feasible set is non-empty.

2.30 Proposition (LP Duality Theorem)

1.) *If either of the primal or dual linear programs has a finite optimal solution, so does the other, and the corresponding values of the objective functions are equal.*

2.) *If either of the primal or dual linear programs has an unbounded objective, the other problem has no feasible solution.*

For a game matrix $M \in \mathbb{R}^{m,n}$ define the matrix $\widetilde{M} \in \mathbb{R}^{m+1,n+1}$ and the vectors $\widetilde{b} \in \mathbb{R}^{m+1}$, $\widetilde{c} \in \mathbb{R}^{n+1}$ by

$$\widetilde{M} = \begin{pmatrix} & & & -1 \\ & M & & \vdots \\ & & & -1 \\ -1 & \cdots & -1 & 0 \end{pmatrix}, \; \widetilde{b} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -1 \end{pmatrix} \; \widetilde{c} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -1 \end{pmatrix}. \tag{2.45}$$

The solution of a matrix game $\Gamma = [M]$ by LP then is treated by the following proposition:

2.31 Proposition (LP for Solving Matrix Games [66])
Let $\Gamma = [M]$ be a matrix game with $V^*(M) > 0$ and let $\widetilde{M}, \widetilde{b}, \widetilde{c}$ as in Equation 2.45, then

[(24)] Other options also exist such as *fictitious play*.
[(25)] Reversely, it is also possible to solve linear programs by special matrix games [208].

1.) the primal and dual LP $(\widetilde{M}, \widetilde{b}, \widetilde{c})$ *are feasible and thus have bounded solutions, and*

2.) the optimal values of both programs equal $-V^*(M)$ *and*

$$\begin{array}{rcl} \pi_2 \in \mathcal{O}^2(M) & \iff & (\pi_2, V^*(M)) \in \mathbb{R}^{n+1} \text{ is optimal in the primal LP.} \\ \pi_1 \in \mathcal{O}^1(M) & \iff & (\pi_1, V^*(M)) \in \mathbb{R}^{m+1} \text{ is optimal in the dual LP.} \end{array}$$

The last row of $[M]$ ensures in the primal LP that $\sum_i (\pi_2)_i \geq 1$ while the last column ensures in the dual LP that $\sum_i (\pi_1)_i \leq 1$. The condition $V^*(M) > 0$ has the effect that these sums equal 1 for an optimal solution and can be replaced by $\pi_1 \in \mathrm{PD}(\mathcal{A}(s_0)),\ \pi_2 \in \mathrm{PD}(\mathcal{O}(s_0))$ [66]. The application of LP can also be seen as a special case ($M_1 = M$, $M_2 = -M$, separate to two independent LPs) of the *Mangasarian-Stone algorithm* for bimatrix games [129] which is the following bilinear quadratic program:

$$\begin{array}{ll} \text{maximise} & \pi_1^T M_1 \pi_2 + \pi_1^T M_2 \pi_2 - (c_1 + c_2) \\ \text{subject to} & \pi_1 \in \mathrm{PD}(\mathcal{A}(s_0)),\ \pi_2 \in \mathrm{PD}(\mathcal{O}(s_0)) \\ & e_{a_i}^T M_1 \pi_2 \leq c_1 \text{ (for all } a_i \in \mathcal{A}(s_0)) \\ & \pi_1^T M_2 e_{o_i} \leq c_2 \text{ (for all } o_i \in \mathcal{O}(s_0)) \\ & c_1, c_2 \in \mathbb{R}, \end{array} \tag{2.46}$$

where $e_{a_i} \in \mathrm{PD}(\mathcal{A}(s_0))$, $e_{o_i} \in \mathcal{O}(s_0))$ are policies corresponding to a pure action, i. e. $(e_{a_i}) = e_i$ being the i-th unit vector. The Mangasarian-Stone algorithm can be suitably enhanced to determine all Nash equilibria in a general (nonzero-sum multi-stage n agent) Markov game [65]. The optimisation criterion as well as the constraints look very similar to the Bellman equation of MDPs (Equations 2.29 and 2.30) which is reasonable because fixing all but one agent's strategies results in an MDP and the Nash equilibrium is defined by the non-motivation of a unilateral deviation of one agent.

2.3.3 Complexity, Algorithmic Issues and Software

The complexity of solving linear programs is in P (polynomial) for interior point methods but for the standard simplex method exponential [208][(26)]. Also numerical studies can be found therein and a bound on the approximation quality of matrix game solutions which was found by von Neumann [155]: To approximate the value $V^*(M)$ of a matrix game $\Gamma = [M]$ with matrix $M \in \mathbb{R}^{m,n}$ by a factor of ε a special linear programming algorithm – different from the simplex method – needs at most

$$\frac{(m+n)(\max_{ij}(M)_{ij} - \min_{ij}(M)_{ij})^2}{\varepsilon^2} \tag{2.47}$$

steps, each of which requires about $4mn$ flops.

Throughout this thesis the solution of matrix games is performed by linear programming methods which are embedded in the MATLAB software package, namely a primal-dual method and a simplex method.

2.4 Two Player Zero Sum Markov Games (2P-ZS-MGs)

Finally, two-player zero-sum Markov games provide a suitable concept for modelling robot soccer. As mentioned before they are a generalisation of both MDPs and matrix games,

[(26)] There exist subexponential variants of the simplex method but they are nevertheless superpolynomial.

for they describe a situation in which two agents try to achieve opposite goals. Despite the generalisation of MDPs, many concepts, statements, and algorithms are transferable from MDPs to 2P-ZS-MGs. For example, the Bellman equation and, therefore, the basic algorithms value iteration and Q-learning only have to changed by determining the solution of a matrix game (max-min) instead of the maximisation problem (max).

One major dissimilarity is the necessity for mixed actions in 2P-ZS-MGs[(27)] while for MDPs it is guaranteed that an optimal policy in pure actions exists [168]. This is induced by the fact that the decisions of both agents at each decision epoch have to be made at the same time without the knowledge of the other agent's decision. The implications are far reaching for more advanced concepts of learning such as the unapplicability of team learning algorithms with coordination mechanisms which rely on a deterministic optimal policy. [103, 90] give some ideas of coordination mechanisms to overcome difficulties with previous approaches [89, 102]. However, applicability of 2P-ZS-MGs includes not only two-person and two-team board games and sports games with competitive character but also the modelling of a worst-case optimal policy in an MDP with uncertainties by considering the "uncertainty generator" as a competitive player (a "game against nature" [147]).

2.4.1 Basics and Problem Definitions

2.32 Definition (Two Player Zero-Sum Markov Game)
A (discrete time, finite) two-player zero-sum Markov game $\mathcal{M} = (\mathcal{D}, \mathcal{S}, \mathcal{SAO}, T, R)$ *is given by*

1.) decision epochs $\mathcal{D} = \mathbb{N}_0$,

2.) *a (finite)* state space $\mathcal{S}$,

3.) *a (finite)* state action space $\mathcal{SAO} = \{(s, a, o) : s \in \mathcal{S}, a \in \mathcal{A}(s), o \in \mathcal{O}(s)\}$ *where* $\mathcal{A}(s), \mathcal{O}(s)$ *are the (finite) sets of available* actions *for the agents* P_1 *and* P_2*, respectively, in state* s,

4.) *a* transition function $T : \mathcal{SAO} \times \mathcal{S} \to [0, 1]$ *with* $T(s, a, o, s')$ *being the probability of reaching state* s' *if choosing the action pair* (a, o) *in state* s,[(28)]

5.) *a (deterministic)* reward function $R : \mathcal{SAO} \to \mathbb{R}$.

Policies π_i are defined for each player as for MDPs. The goal of determining an optimal policy for agent P_1 and the definitions of *value functions* V and *returns* $\mathcal{R}$ are (nearly) the same as for MDPs.[(29)] The reason why the value function and return are independent from the policy π_2 of the second agent is that it always is assumed to be a worst-case answer against the first agent ($\pi_2 \in BR(\pi_1)$). Hence, the second player's policy is not an independent variable but depends on the policy of the first one: $\pi_2 = \pi_2(\pi_1)$. Theorem 2.20 yields the well-definedness of the optimal value function: the zero-sum property guarantees that for all $\pi_2^* \in BR(\pi_1^*)$ the value functions and returns are equal. [159] contains a proof of existence of a pair of (stationary) optimal policies for 2P-ZS-MGs.

As alluded to in the introduction the max-operator simply needs to be replaced by a (max-min)-operator in the Bellman equation of an MDP (Equation 2.29) in order to obtain the corresponding optimality principle for a 2P-ZS-MG:

[(27)] [112] refers to rock-paper-scissors as an example for the need of *mixed actions* in 2P-ZS-MGs.
[(28)] An alternative for 2P-ZS-MGs with deterministic state transitions is to define the modified transition function $\widetilde{T} : \mathcal{SAO} \to \mathcal{S}$ with $T(s, a, o) = s'$ being the next state which is reached with probability 1.
[(29)] Especially, the discounted return $\mathcal{R}_{\text{disc}}$ is considered to be standard.

2.33 Theorem (Optimality Principle (Shapley's Theorem [66]))
A (state-) value function V of an 2P-ZS-MG is optimal iff it is the unique solution to the Bellman equation:

$$V(s) = \max_{\pi_s \in \mathrm{PD}(\mathcal{A}(s))} \min_{o \in \mathcal{O}(s)} \sum_{a \in \mathcal{A}(s)} \pi_s(a) \cdot Q(s,a,o) \tag{2.48}$$

where

$$Q(s,a,o) = R(s,a,o) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,o,s') \cdot V(s'). \tag{2.49}$$

As for MDPs, the result of plugging Equation 2.49 into Equation 2.48 can be abbreviated by the *Bellman operator* $\mathcal{B}_{\mathrm{MG}}$ which shortens the notation of the Bellman equation to $V = \mathcal{B}_{\mathrm{MG}} V$ and the operator is a contraction with rate γ with respect to $\| \, \|_\infty$ [97]. Again, the Bellman equation is a non-linear fixed point equation for the optimal value function $V^* = V^{\pi^*}$.

One additional result which is not mentioned for MDPs but is also valid because these are special cases of 2P-ZS-MGs concerns the continuous dependency of the optimal value function on the Markov game:

2.34 Theorem (Continuity of Optimal Value Functions [66])
Given a state space $\mathcal{S}$ and a state-action space $\mathcal{SAO}$, the optimal value function $V^* = V^*(R,T,\gamma)$ is continuous with respect to the metric

$$d_{R,T,\gamma}((R_1,T_1,\gamma_1),(R_2,T_2,\gamma_2)) = \max\{ \, \|R_1 - R_2\|_R, \, \|T_1 - T_2\|_T, \, |\gamma_1 - \gamma_2| \, \} \tag{2.50}$$

with

$$\|R_1 - R_2\|_R = \max_{(s,a,o)} |R_1(s,a,o) - R_2(s,a,o)| \tag{2.51}$$

and

$$\|T_1 - T_2\|_T = \max_{(s,a,o)} \sum_{s' \in \mathcal{S}} |T_1(s,a,o,s') - T_2(s,a,o,s')|. \tag{2.52}$$

2.4.2 Numerical Methods

As for MDPs, the two important classes of approaches are *dynamic programming* (DP) and *reinforcement learning* (RL) methods. Most of the basic methods can also be applied to 2P-ZS-MGs including value iteration [159], policy iteration (without finite time convergence), and Q-learning [203]. The reason is that not only the max-operator is a *non-expansion* (i. e. $|\max_a f(s,a) - \max_a g(s,a)| \leq \max_a |(f(s,a) - g(s,a)|$ for all functions f, g and all states s, [203]) but also the max-min: $|\max_{\pi_a} \min_o f(s,a,o) - \max_{\pi_a} \min_o g(s,a,o)| \leq \max_{(a,o)} |f(s,a,o) - g(s,a,o)|$. The statement for the max-operator follows from the fact that, if without restriction $\max_a f(s,a) \geq \max_a g(s,a)$ and $a^* = \arg\max_a f(s,a)$, then $|\max_a f(s,a) - \max_a g(s,a)| = f(s,a^*) - \max_a g(s,a) \leq f(s,a^*) - g(s,a^*) \leq \max_a |f(s,a) - g(s,a)|$. Regarding the max-min, the non-expansion property is stated already in Proposition 2.25, because the max-min of $f(s,a,o)$ is the value of a matrix game with action pairs (a,o) (for each f and each s).

The definition of value iteration is the only which is repeated in this section to stress the similarities: the difference is in fact only to replace $\mathcal{B}_{\mathrm{MDP}}$ by $\mathcal{B}_{\mathrm{MG}}$. According to [21] the value iteration algorithm for 2P-ZS-MGs was already designed by Shapley [186] before the corresponding algorithm for MDPs.

2.35 Definition (Value Iteration (2P-ZS-MG))
The following algorithm is called value iteration*: select $\varepsilon > 0$, choose an arbitrary initial guess $V_0 \in \mathbb{R}^{|\mathcal{S}|}$ for the (state-) value function, and determine iteratively $V_k = \mathcal{B}_{\mathrm{MG}} V_{k-1}$ for $k = 1, 2, \ldots$ until $\|V_{k+1} - V_k\|_\infty \leq \frac{\varepsilon}{2} \cdot \frac{1-\gamma}{\gamma}$.*

This algorithm *would* converge to V^*, and provide an $\frac{\varepsilon}{2}$-approximation for the value function estimate and an ε-optimal stationary policy (as for MDPs), *if* one assumes that $\mathcal{B}_{\mathrm{MG}}$ can be calculated exactly. However, as proposed in Section 2.3 the solution of the matrix game by LP methods can only be numerically determined. Although for numerically applying $\mathcal{B}_{\mathrm{MDP}}$ the maximum of finitely many values can exactly be determined, the representation of real numbers by machine numbers may lead to small discrepancies.

In the following, the result of Lemma 4.2 is anticipated since it fits well into this context and its interpretation is new in the sense that the error of solving a matrix game is mathematically equivalent to the error of using supervised learning techniques. Also, the comments and discussion in Section 4.2 are correspondingly valid.

2.36 Lemma (Error of Numerical Value Iteration (2P-ZS-MG))
Let $\mathcal{B}_{\mathrm{MG}}$ be the Bellman operator and $\widetilde{\mathcal{B}}_{\mathrm{MG}}$ a numerical realisation with $\|\widetilde{\mathcal{B}}_{\mathrm{MG}} V - \mathcal{B}_{\mathrm{MG}} V\|_\infty \leq \varepsilon_1 \|V\|_\infty$.[30] *Let further be $\widetilde{V}_0 = V_0$, $V_k = (\mathcal{B}_{\mathrm{MG}})^k V_0$, and $\widetilde{V}_k = (\widetilde{\mathcal{B}}_{\mathrm{MG}})^k \widetilde{V}_0$ the corresponding k-th value iterates. Then*

$$\|\widetilde{V}_k - V_k\| \leq \underbrace{\varepsilon_1 \cdot \sum_{i=0}^{k-1} \gamma^{k-i-1} \|\widetilde{V}_i\|_\infty}_{=\mathcal{E}_V(k)} \leq \frac{\varepsilon_1}{1-\gamma} \max_{i=0,\ldots,k-1} \|\widetilde{V}_i\|_\infty \,. \tag{2.53}$$

2.37 Corollary (New Stopping Criterion for Numerical Value Iteration)
If the stopping criterion is changed to $\|\widetilde{V}_{k+1} - \widetilde{V}_k\| \leq c(\widetilde{V}_0, \ldots, \widetilde{V}_k)$ with

$$c(\widetilde{V}_0, \ldots, \widetilde{V}_k) = \Big(\frac{\varepsilon}{2} - \mathcal{E}_V(k+1)\Big) \cdot \frac{1-\gamma}{\gamma} \;-\; (\mathcal{E}_V(k+1) + \mathcal{E}_V(k)) \tag{2.54}$$

where the errors $\mathcal{E}_V(k)$ depend on the first $k-1$ numerical value iterates (Equation 4.3), and secondly, if the numerical approximation $\widetilde{\mathcal{B}}_{\mathrm{MG}}$ is a contraction of rate γ (like $\mathcal{B}_{\mathrm{MG}}$) then also the numerical approximation of value iteration yields results comparable to the original value iteration.

2.4.3 Complexity, Algorithmic Issues and Software

In principle, the statements for MDPs of Section 2.2.3 are valid e. g. for the effort per iteration with the difference that the costs of solving $|\mathcal{S}|$ matrix games of possibly different sizes $|\mathcal{A}(s)| \cdot |\mathcal{O}(s)|$ have to be added. The complexity of solving matrix games e. g. by LP is discussed in Section 2.3.3. Furthermore, Condon shows for the case of simple stochastic games, which are a restricted class of two-player games, that to decide whether the probability of winning for one player is > 0.5 is in NP $\cap$ co-NP [33].

[30] $\|\widetilde{\mathcal{B}}_{\mathrm{MG}} V - \mathcal{B}_{\mathrm{MG}} V\|_\infty \leq \varepsilon_1 \|V\|_\infty$ for all V means that the operator $\mathcal{B}_1 := \frac{1}{\varepsilon_1}(\widetilde{\mathcal{B}}_{\mathrm{MG}} - \mathcal{B}_{\mathrm{MG}})$ has operator norm less than 1: $\|\mathcal{B}_1\| = \sup_{V \neq 0} \frac{\|\mathcal{B}_1 V\|_\infty}{\|V\|_\infty} \leq 1$. Furthermore, the above definition implies $\widetilde{\mathcal{B}}_{\mathrm{MG}} = \mathcal{B}_{\mathrm{MG}} + \varepsilon_1 \mathcal{B}_1$.

There are free available software frameworks for reinforcement learning but typically they are only designed to solve MDPs. Often, it is time consuming to provide a transition model and a suitable representation of the problem. Thus, the MATLAB based software package DRPOST is implemented by the author for the special use of determining optimal strategies in multi-player grid soccer.

2.5 General Markov Games, Differential Games, and Advanced Concepts of RL

In this section pointers to interesting concepts that are related to the previous part of the section but are not a main focus of this thesis shall be given. Some of the concepts show alternatives to those in Section 5.1 of modeling a robot soccer game. The range of concepts reaches from general n-player games via differential games to more advanced concepts of learning.

Discrete Games with n Agents

It can not be expected that the results of this thesis are extendable to *general-sum multi-player Markov games* as [218] shows that suitably adapted value iteration methods (best response instead of max-min strategies) do not even need to converge to a stationary policy. This is even true in games with two players, alternating turns, and deterministic transitions.

Differential Games

In contrast to other games discussed before differential games model time as being continuous (RL with continuous time is treated by [56]). Two standard textbooks on differential games are [86] and, especially on pursuit-evasion games, [107] where some of the notation is adapted from.

The evolution of a differential game is modeled by a differential equation

$$\dot{x} = f(x, u, v), \tag{2.55}$$

where $\dot{x} = \frac{\mathrm{d}x}{\mathrm{d}t}$ and $x(t), u(t), v(t) \in \mathbb{R}^n$. This can also be seen as a dynamical system with two controls u, v. Other aspects which are necessary to describe a game is a terminal function which determines whether a terminal state of the game is reached (often one considers merely a fixed terminal set), and an outcome functional of the game for every player P_i that can consist of an outcome dependent on a trajectory and a final outcome dependent on the final state. The aim of each player is – as for MDPs – to maximise its own outcome. Some statements about uniqueness and continuity of the solution to differential game trajectories including different information patterns can be found in [13].

In a newer context, *viscosity solutions* to the so-called *Hamilton-Jacobi-Bellman-Isaacs equation* (non-linear PDE) are introduced, see [9, 36] and references therein. This concept may be considered a weak solution concept which overcomes the difficulty that because of the non-differentiability of the value functions, this function is no solution to the Hamilton-Jacobi-Bellman-Isaacs equation and other generalisations may yield non-unique solutions (even for MDPs [152]). A good overview of theoretical results as well as numerical approximation schemes and examples is contained in [63].

The concept of generalisation is discussed at more length in Chapter 4; model reduction is outlined in Chapter 3. At this place, semi-Markov decision processes (S-MDPs), hierarchical Markov decision processes (H-MDPs), partial observability (PO-MDPs), and some trade-offs and special artifices to speed up convergence are to be addressed.

Hierarchical learning and S-MDPs. S-MDPs are a generalisation of MDPs with respect to the duration of performing actions. The assumption of MDPs that every time step is equidistant (every action takes the same amount of time) is dropped for S-MDPs. Continuous-time discrete event versions exist as well as the easier discrete time version which allows only durations being integer multiples of a unit duration. S-MDPs are the natural framework for H-MDPs [11] which typically consist of a hierarchy of learning problems with at least two different levels. Dietterich [53] provides with MAX-Q a multi-level approach in which the hierarchy structure must be given in advance. [11] gives a broad overview. Earlier examples often follow a two-level approach and include [42, 55, 109, 124, 127, 190] while newer examples include [81, 161, 197, 215]. [191] gives additional useful references.

Partially observable MDPs (PO-MDPs). One problem of real world tasks is often that the observed information is noisy or incomplete. Unfortunately, complete information is necessary to solve MDPs because the agent needs to know the state for which the updates of the value function have to be done. A formal model which incorporates the effect of a lack of information is called *partially observable Markov decision process*. There are several strategies to deal with PO-MDPs [88]: State-free deterministic or stochastic policies i. e. ignoring the partial observability can lead to non-Markovity. Determining a deterministic optimal policy (mapping from observation to actions) is NP-hard [113]. By stochastic policies locally optimal results can be obtained [87]. True improvements can only be observed in most environments by memorising previous actions and observations. Some approaches are *recurrent Q-learning* (using a recurrent neural network to learn features of history [110, 137, 182]), *classifier systems* [71, 54, 27, 100] which were originally similar to Q-learning, interval-based estimation of the transition probabilities [38], and finite-history window approaches with fixed [110] or variable length (*utile suffix memory*) [134], possibly in combination with neural network approaches [172]. Finally, PO-MDP approaches use a *hidden Markov model* (HMM) to learn a model of the environment and construct a *perfect memory controller* [29, 119, 140, 173, 174, 189]. The discrete PO-MDP can be transformed into an equivalent continuous space *belief MDP*, with the probabilities distributions over the discrete states being the new states. Thus, roughly speaking, an PO-MDP with n states can be converted to an MDP with state space $\subseteq \mathbb{R}^n$ which clearly shows the limits of this approach. Principal component analysis seems to be an appropriate method to reduce the huge belief space of PO-MDPs [174]. A good overview of PO-MDPs and a comparison of methods for *very small* problems (less than 20 states, 5 actions, and 10 observation states) can be found in [114].

Special Artifices to Speed Up Convergence of DP and RL methods. Convergence properties are in general asymptotic i. e. to some degree useless for practical needs where the convergence rate at the beginning of learning is more interesting. Therefore, speed of convergence is an ill-defined measure while speed of convergence to near-optimality is more fruitful [88]. One performance measure in this sense is the *regret* [17] which describes the loss of return while learning a policy in comparison to executing the (a priori known) optimal policy. Of course, it is very hard to estimate the regret. Other artifices are updating

states with higher *Bellman error* ($\|V_{k+1} - V_k\|_\infty$) more often, using combinations of value iteration and policy iteration like methods (general policy iteration [202]), modifying the model (mostly the reward function (shaping) or the start position (Q-learning)), or using heuristic knowledge to avoid starting from scratch (modify Q_0, V_0 such that it corresponds to a heuristic (human) policy).

Learning versus Dynamic Programming

The advantage of RL methods is that they do not need a model while the disadvantage is that they need a lot of training data or episodes to create reliable estimates. For the latter example of soccer a middle course shall be followed: an offline computation with a coarse model yields a good estimate of the value function. In the real game, this estimate will be the initial value function from which the learning process starts instead of starting from scratch. One goal of this thesis is to construct such an initial guess from a coarse model for a competitive multi-agent soccer game.

Two further issues, which are especially under consideration in RL but also are related to DP, are the *temporal credit assignment problem* which concerns the question how much a single action contributes to the complete sequence of actions, and the *structural credit assignment problem* of cooperative multiple agents, i. e. how much a single agent contributes to solving the overall task [1]. A survey over cooperative agents with exhaustively many references is [160].

Chapter 3

Model Reduction and Symmetry

Contents

> Order and simplification are the first steps
> towards the mastery of a subject.
>
> P. Thomas Mann (1875–1955)
> (http://www.nonstopenglish.com)

Symmetries which are an essential source for model reduction address two important issues: first, it can generally be considered sensible that any two models describing the same situation, e. g. the discretisation of a continuous model, should have the same symmetries, and second, the abstraction over equivalent state(-action)s implies faster algorithms e. g. faster learning comparable to the spirit of function approximation.(1)

Temporal and structural abstraction are main challenges in solving real world MDPs and 2P-ZS-MGs [67]. The latter concerns the question of how to handle the state space of the underlying model for very large problems. In this context, model reduction, i. e. finding an equivalent smaller or *the* smallest model, can be seen to be an important part of the task to make computations more effective or simply feasible. While some reductions, especially symmetries, are often easy to detect for a human observer, others are harder to observe. The design of a software to accomplish this task is a challenging area of research for any kind of model reduction.(2)

In this chapter proofs are given that some kinds of reduction yield models equivalent to the original one and, hence, the effort to determine a reduced model can be valuable. A

(1) Even if a symmetry is not noted in advance, approximating functions which respect this symmetry should be better suited than others.

(2) A complexity result of [70] is that it is NP-hard to determine whether a given finite model is already minimal under reduction by symmetry groups (see Section 3.1.2).

second important point, namely, how to compute possible reductions will not be addressed in detail. One possible method is adaptive state aggregation (clustering of states) by locality or by the similarity of the value function [216][(3)]; a different one is to detect steep gradients of the value function as a natural barrier between state clusters [58][(4)]. Optimal value functions for multiple goals can also be used to uncover structure in the state space [67].[(5)] However, all these methods typically yield only *approximate* reductions whereas this chapter concerns *exact* reductions.

The chapter is structured as follows: after some remarks about existing work for MDPs on MDP homomorphisms and MDP symmetries the equivalence of these concepts is proven by the author in Section 3.1. Furthermore, the previous concepts are related to group actions in Section 3.1.2. In Section 3.2, the focus will be on extensions to 2P-ZS-MGs which is also one of the main theoretical contributions of this thesis. A central aspect in 2P-ZS-MGs, which is not an issue in MDPs, is the matrix game reduction property (Definition 3.12) which is proven to be valid if the equivalence classes on the state-action space fulfill some natural projection properties. The most general framework of model reduction leads to 2P-ZS-MG μ-homomorphisms (Theorem 3.20). The fact that the composition of two 2P-ZS-MG μ-homomorphisms stays a 2P-ZS-MG μ-homomorphism enables the stepwise reduction of models. Additionally, the combination of agent exchanging symmetries with agent preserving symmetries is part of the concept. An application to automorphisms of finite 2P-ZS-MGs (Lemma 3.25) gives some further insight.

3.1 Homomorphisms and Symmetry in MDPs

The concept of symmetries is strongly related to questions of model reduction and model minimisation. Historically, the question of model minimisation emerged first in finite state automata and was extended to Markov chains and MDPs [170, 171]. The motivation of this work is the insight that a smaller model results in a smaller amount of computation time since the computational complexity typically depends on the model size (compare Section 2.2.3 and Remark 3.7). The fundamental algebraic elements are homomorphisms between two different models which represent the idea of equivalence of these two models. For an equivalence of two MDPs not only does the structure of the state(-action) space have to be preserved by a homomorphism but also the structure of the transition and reward functions, implying the same structure for the optimal value functions and optimal policies.

This section provides background material for Section 3.2 but also contains theoretical contributions of the author. A main contribution is to show the equivalence of MDP homomorphisms [171] to MDP symmetries [217] in Lemma 3.3.[(6)] This equivalence reveals that the model minimisation framework of [171] with MDP options and the symmetry context of [217] with its generalisation to multi-agent MDPs do not exclude each other but

[(3)] The presented approach has the burden that the value function of non-aggregated states also needs to be stored and that there are an exponential number of cluster intern (deterministic) policies over which a maximum is searched for.

[(4)] This approach seems to be most appropriate for detection of walls and implicit barriers.

[(5)] The problem with this approach is that one needs to calculate some value functions for different goals before one can benefit from the structural abstraction.

[(6)] In principle, many results of this section can be found in the technical report [170] but there are e. g. no pointers to the relation to group actions nor any note – if noticed – that the concept of reward respecting SSP partitions in fact is the MDP symmetry of Zinkevich. The main reason for this may be that some conditions or implications are left implicit in the technical report which are made explicit by the author.

instead are based on the same foundations. Furthermore, because of the equivalence it is sufficient to prove only once e. g. that the optimal value function is constant on equivalence classes induced by MDP homomorphisms which also implies that this is true for equivalence classes induced by MDP symmetries.

3.1.1 Equivalence of MDP Homomorphisms and MDP Symmetries

MDP homomorphisms have the concept of preserving some structure in common with group homomorphism. For MDPs this includes maps for states, state-actions, transition functions, and reward functions. Because MDP homomorphisms shall be utilised to model reduction, the original MDP $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SA}_1, T_1, R_1)$ has a larger number of states or state-actions than the reduced MDP $\mathcal{M}_2 = (\mathcal{D}_2, \mathcal{S}_2, \mathcal{SA}_2, T_2, R_2)$. In this context, an MDP homomorphisms *aggregates some states* and state-actions, i. e. merges them into non-trivial equivalence classes.

Before MDP homomorphisms can be defined some aspects of projecting partitions (or equivalence classes [60]) of the state-action space onto partitions of the state space have to be noted. A first assumption in [171] is that a partition of the (finite) state-action space $\mathcal{P}(\mathcal{SA}_1) = \{\, [(s,a)] : (s,a) \in \mathcal{SA}_1 \}$ into equivalence classes is given. It is further assumed that this partition induces a partition of equivalence classes on the state space $\mathcal{P}(\mathcal{S}_1) = \{\, [s] : s \in \mathcal{S}_1 \}$ by the direct *projection* $\Pi_s : \mathcal{SA}_1 \to \mathcal{S}_1$ with $\Pi_s(s_1, a_1) = s_1$ by means of $[s] = \Pi_s([(s,a)])$. The well-definedness of this projection onto state equivalence classes is equivalent to the condition

$$\forall (s,a'') \in \mathcal{SA}_1 \; \forall (s,a) \in \mathcal{SA}_1 \; \forall s' \in \Pi_s([(s,a'')]) \; \exists a' \in \mathcal{A}_1(s') : \; (s',a') \in [(s,a)] \,. \tag{3.1}$$

If the above equation is fulfilled (which is assumed by [171]) then it can be rewritten in terms of the induced equivalence classes because $[s] = \Pi_s([(s,a)]) = \Pi_s([(s,a'')])$:

$$\forall (s,a) \in \mathcal{SA}_1 \; \forall s' \in [s] \; \exists a' \in \mathcal{A}_1(s') : \; (s',a') \in [(s,a)] \,. \tag{3.2}$$

The well-definedness of the projection of equivalence classes by Π_s further implies (not especially noted in [171]) the following condition on the equivalence classes:

$$\forall (s,a) \in \mathcal{SA}_1 \; \forall (s',a') \in \mathcal{SA}_1 : \; \Big((s',a') \in [(s,a)] \Rightarrow s' \in [s] \Big) \,. \tag{3.3}$$

These two conditions (Equations 3.2 and 3.3) are also needed in the symmetry formalism of Zinkevich and Balch: they simply are Definition 7 of [217] with $[(s,a)]$ being the equivalence classes of the relation E_{SA} and $[s]$ being the equivalence classes of E_S in their notation. Considering the reverse direction, i. e. if equivalence relations are given on $\mathcal{S}_1$ and $\mathcal{SA}_1$ which satisfy Equations 3.2 and 3.3, then the projection Π_s exactly maps equivalence classes from $\mathcal{SA}_1$ to the ones of $\mathcal{S}_1$. The reason is that Equation 3.2 guarantees that the equivalence classes on $\mathcal{S}_1$ are small enough to be induced by the projection Π_s, while Equation 3.3 assures that the equivalence classes on $\mathcal{S}_1$ are also large enough.

Next, MDP homomorphisms can be defined:

3.1 Definition (MDP Homomorphism [171])
Let $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SA}_1, T_1, R_1)$ and $\mathcal{M}_2 = (\mathcal{D}_2, \mathcal{S}_2, \mathcal{SA}_2, T_2, R_2)$ be two MDPs as defined in Section 2.2 with $\mathcal{D}_1 = \mathcal{D}_2 = \mathbb{N}_0$. A map $h : \mathcal{SA}_1 \to \mathcal{SA}_2$ is called an MDP homomorphism if h is a surjection, defined by a tuple of surjective maps $(f, (g_s)_{s \in \mathcal{S}})$ with $h(s,a) = (f(s), g_s(a))$, $f : \mathcal{S}_1 \to \mathcal{S}_2$, and $g_s : \mathcal{A}_1(s) \to \mathcal{A}_2(f(s))$ such that

$$\forall (s,a) \in \mathcal{SA}_1 \; \forall s' \in \mathcal{S}_1 : \; \widetilde{T}_1(s,a,[s']) = T_2(f(s), g_s(a), f(s')) \tag{3.4}$$

and

$$\forall (s,a) \in \mathcal{SA}_1 : \; R_1(s,a) = R_2(f(s), g_s(a)) \tag{3.5}$$

where the block transition function $\widetilde{T}_1$ *of the MDP* $\mathcal{M}_1$ *is defined by*

$$\widetilde{T}_1 : \mathcal{SA}_1 \times \{[s] : s \in \mathcal{S}_1\} \to \mathbb{R}, \; \widetilde{T}_1(s,a,[s']) = \sum_{s'' \in [s']} T_1(s,a,s'')^{(7)} \tag{3.6}$$

and the equivalence classes $[(s,a)]$ *on* $\mathcal{SA}_1$ *are defined by* $(s',a') \in [(s,a)]$ *iff* $h(s',a') = h(s,a)$ *and the equivalence classes* $[s]$ *on* $\mathcal{S}_1$ *are defined by the projections* $\Pi_s([(s,a)])$ *which makes Equation 3.1 a necessary condition.*[(8)]

To provide the material necessary for a comparison between the MDP homomorphisms of [171] and the equivalence relation notation of [217], the latter one remains to be introduced. On that account, the notion of an equivalence relation E on a set M which is a subset $E \subseteq M \times M$ with the properties reflexivity ($(x,x) \in E$), symmetry ($(x,y) \in E \Rightarrow (y,x) \in E$), and transitivity ($(x,y) \in E$, $(y,z) \in E \Rightarrow (x,z) \in E$) is to be recalled. An equivalence relation gives rise to the *quotient set* $M/E = \{B \subset E \mid (x,y) \in E \text{ for all } x,y \in B\}$ of equivalence classes of M by E. As above the notation $x \in [y]$ means that x is in the equivalence class of y.

In the following definition the notation of [217] is slightly changed to stress the similarities to MDP homomorphisms:

3.2 Definition (MDP Symmetry [217])
Let $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SA}_1, T_1, R_1)$ *be an MDP. An MDP symmetry is a tuple* $E = (E_{\mathcal{S}_1}, E_{\mathcal{SA}_1})$ *of equivalence relations on* $\mathcal{S}_1$ *and* $\mathcal{SA}_1$*, respectively, such that for the corresponding equivalence classes* $[s]$ *and* $[(s,a)]$ *Equations 3.2 and 3.3 are valid, and additionally*

1.) the block transition function $\widetilde{T}_1$ *defined by Equation 3.6 is constant on equivalence classes:*

$$\forall (s',a') \in [s,a] \; \forall s'' \in \mathcal{S}_1 : \; \widetilde{T}_1(s',a',[s'']) = \widetilde{T}_1(s,a,[s'']) \,, \tag{3.7}$$

2.) and the reward function is constant on equivalence classes:

$$\forall (s',a') \in [s,a] : \; R_1(s',a') = R_1(s,a) \,. \tag{3.8}$$

3.3 Lemma (Equivalence of MDP Homomorphisms and MDP Symmetries)
MDP homomorphisms (Definition 3.1) and MDP symmetries (Definition 3.2) are equivalent, i. e. for each MDP homomorphisms there exists an MDP symmetry and for each MDP symmetry there exists an MDP homomorphism such that the equivalence classes induced by the MDP homomorphism and the MDP symmetry are equal.[(9)]

Proof: It is to show firstly that given any MDP symmetry there exists an equivalent MDP homomorphism and, secondly, that the opposite direction of implication is also true. For

[(7)] The block transition function is in fact a transition function because $\forall (s,a) \in \mathcal{SA}_1 : \sum_{[s']} \widetilde{T}_1(s,a,[s']) = \sum_{[s']} \sum_{s'' \in [s']} T_1(s,a,s') = \sum_{s' \in \mathcal{S}_1} T_1(s,a,s') = 1$ because T_1 is a transition function.

[(8)] An implication is that not only the equivalence classes on the state(-action) space can be written as $[(s,a)] = h^{-1}(h(s,a))$ but also $[s] = f^{-1}(f(s))$.

[(9)] If an equivalence relation on all MDP homomorphisms is introduced such that two of them are equivalent if the induced equivalence relation on the state and state-action spaces are equal, and if similarly an equivalence relation of MDP symmetries is introduced, then the Lemma means that a bijection between equivalence classes of MDP homomorphisms and equivalence classes of MDP symmetries exists.

clearness of the presentation equivalence classes induced by an MDP homomorphism h are denoted by $[s]_h$ and $[(s,a)]_h$ while the ones induced by an MDP symmetry $E = (E_{\mathcal{S}_1}, E_{\mathcal{SA}_1})$ are described by $[s]_{E_{\mathcal{S}_1}}$ and $[(s,a)]_{E_{\mathcal{SA}_1}}$.

1.) Let an MDP $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SA}_1, T_1, R_1)$ and an MDP symmetry $E = (E_{\mathcal{S}_1}, E_{\mathcal{SA}_1})$ be given. Above Definition 3.1 it is discussed that Equations 3.2 and 3.3 imply the well-definedness of the projection $\Pi_s([(s,a)]_{E_{\mathcal{SA}_1}}) = [s]_{E_{\mathcal{S}_1}}$ from equivalence classes of $\mathcal{SA}_1$ onto that of $\mathcal{S}_1$. Because of this well-definedness it is possible to choose a unique (finite) set of representatives $N_{s,a} = \{(s_i, a_i)\}$, one for each equivalence class on $\mathcal{SA}_1$, such that $N_s = \Pi_s(N_{s,a}) = \{s_i\}$ is a unique (finite) set of representatives for the equivalence classes on $\mathcal{S}_1$. Be $\Phi_{s,a} : \mathcal{SA}_1 \to N_{s,a}$ the mapping which maps a state-action (s,a) to its representative $(s_k, a_k) \in [(s,a)]_{E_{\mathcal{SA}_1}} \cap N_{s,a}$ and $\Phi_s : \mathcal{S}_1 \to N_s$ the corresponding mapping for the state space[(10)].
Let a second MDP $\mathcal{M}_2 = (\mathcal{D}_2, \mathcal{S}_2, \mathcal{SA}_2, T_2, R_2)$ be defined by $\mathcal{D}_2 = \mathcal{D}_1$, $\mathcal{S}_2 = N_s$, $\mathcal{SA}_2 = N_{s,a}$, $T_2(s,a,s') = \widetilde{T}_1(\Phi_{s,a}(s,a), [\Phi_s(s')]_{E_{\mathcal{S}_1}})$, and $R_2(s,a) = R_1 \circ \Phi_{s,a}(s,a)$. Then, an MDP homomorphism $h(s,a) = h(f(s), g_s(a))$ is defined by $h(s,a) = \Phi_{s,a}(s,a)$, implying that $f(s) = \Phi_s(s)$ and $g_s(a) = \Pi_a(\Phi_{s,a}(s,a))$ where Π_a is the direct projection on the actions and $\mathcal{A}_2(f(s)) = \{\Pi_a(\Phi_{s,a}(s,a)) : \ a \in \mathcal{A}_1(s))\}$ which is independent from the representative $f(s) = \Phi_s(s)$ of $[s]_{E_{\mathcal{S}_1}}$ by means of Equation 3.2.
For the proof that h is an MDP homomorphism the first step is to note that the equivalence classes induced by h on $\mathcal{SA}_1$, i. e. $(s',a') \in [(s,a)]_h$ iff $h(s',a') = h(s,a)$ are by construction the same as that of $E_{\mathcal{SA}_1}$ i. e. $[(s,a)]_h = [(s,a)]_{E_{\mathcal{SA}_1}}$. Because Π_s is well-defined for the $[(s,a)]_{E_{\mathcal{SA}_1}}$ equivalence classes it is also well-defined for the $[(s,a)]_h$ cases, hence $[s]_h = \Pi_s([(s,a)]_h) = \Pi_s([(s,a)]_{E_{\mathcal{SA}_1}}) = [s]_{E_{\mathcal{S}_1}}$ and especially Equation 3.1 is valid for $[s]_h$ and $[(s,a)]_h$.
The second step is to show that Equations 3.4 and 3.5 hold. Equation 3.5 directly follows because the reward function is constant on equivalence classes of $E_{\mathcal{SA}_1}$ (Equation 3.8) and therefore also on equivalence classes of h which are characterised by $(s',a') \in [(s,a)]_h$ iff $(f(s), g_s(a)) = h(s,a) = h(s',a') = (f(s'), g_{s'}(a'))$. Equation 3.4 analogously follows hence the function $(s,a) \mapsto \widetilde{T}_1(\Phi_{s,a}(s,a), \Phi_s(s'))$ is constant on equivalence classes of $E_{\mathcal{SA}_1}$ (Equation 3.7) and therefore also on that of h.

2.) Now, let an MDP homomorphism $h : \mathcal{SA}_1 \to \mathcal{SA}_2$ be given for the two MDPs $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SA}_1, T_1, R_1)$ and $\mathcal{M}_2 = (\mathcal{D}_2, \mathcal{S}_2, \mathcal{SA}_2, T_2, R_2)$ with $\mathcal{D}_1 = \mathcal{D}_2 = \mathbb{N}_0$. The discussion above Definition 3.1 yields that Equations 3.2 and 3.3 are valid for the equivalence classes $[(s,a)]_h$ and $[s]_h$ induced by h. Define equivalence relations $E_{\mathcal{S}_1}$ by $[s]_{E_{\mathcal{S}_1}} = [s]_h$ and $E_{\mathcal{SA}_1}$ by $[(s,a)]_{E_{\mathcal{SA}_1}} = [(s,a)]_h$.
By definition the equivalence classes induced by h and the ones induced by $E = (E_{\mathcal{S}_1}, E_{\mathcal{SA}_1})$ are equal and the latter ones also fulfill Equations 3.2 and 3.3. Therefore, in the opposite direction Equation 3.8 directly follows from Equation 3.5 and Equation 3.7 from Equation 3.4 which means that $E = (E_{\mathcal{S}_1}, E_{\mathcal{SA}_1})$ is an MDP symmetry.

□

3.4 Remark (Implications of Lemma 3.3)
A direct implication of Lemma 3.3 is that the work of [217] and [171] has a common basis. The concepts particularly in the first reference for multi-agent systems and in the second reference for options, i. e. macro or multi-step actions, may be jointly used. As a further

[(10)] The function $\Phi_s : \mathcal{S}_1 \to N_s$ can also be defined by $\Phi_s(s) = \Pi_s(\Phi_{s,a}(s, a(s)))$ where for each s any $a = a(s) \in \mathcal{A}_1(s)$ can be chosen.

consequence, the proofs about value functions and strategies being equal on equivalence classes need only be given in one framework.

After having established the equivalence of both frameworks their usefulness is to be reflected in a Theorem. Before the theorem can be stated, an idea has to be given how to transform policies from $h(\mathcal{SA}_1)$ to $\mathcal{SA}_1$ if h is an MDP homomorphism:

3.5 Definition (Policy Lifting [171])
Let $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SA}_1, T_1, R_1)$ and $\mathcal{M}_2 = (\mathcal{D}_2, \mathcal{S}_2, \mathcal{SA}_2, T_2, R_2)$ be two MDPs and let $h : \mathcal{SA}_1 \to \mathcal{SA}_2$, $h(s,a) = (f(s), g_s(a))$ be an MDP homomorphism. For any $s \in \mathcal{S}_1$ and $\widehat{a} \in \mathcal{A}_2(f(s))$ the action space $g_s^{-1}(\widehat{a}) \subseteq \mathcal{A}_1(s)$ is the preimage of the action $\widehat{a}$ under g_s. Let $\widehat{\pi}$ be a policy of the MDP $\mathcal{M}_2$. Then the corresponding lifted policy π of the MDP $\mathcal{M}_1$ is defined by

$$\pi(s,a) = \frac{\widehat{\pi}(f(s), g_s(a))}{|g_s^{-1}(g_s(a))|} \,. \tag{3.9}$$

The policy lifting simply means to assign the same fraction of probability to all actions in the same state-action equivalence class. The main theorem about MDP homomorphisms (and MDP symmetries) follows:

3.6 Theorem (Main Implications of MDP Homomorphisms [170])
Let $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SA}_1, T_1, R_1)$ be an MDP and let $h : \mathcal{SA}_1 \to \mathcal{SA}_2$, $h(s,a) = (f(s), g_s(a))$, be an MDP homomorphism (Definition 3.1) for some MDP $\mathcal{M}_2 = (\mathcal{D}_2, \mathcal{S}_2, \mathcal{SA}_2, T_2, R_2)$. Then, $V^*(s) = V^*(f(s))$ and $Q^*(s,a) = Q^*(h(s,a))$. Furthermore, there exists an optimal policy of $\mathcal{M}_1$ which is a lifted optimal policy of $\mathcal{M}_2$.

The proof will be given in Section 3.2 for 2P-ZS-MGs which is methodically similar and includes MDPs.

3.7 Remark (Implications of Theorem 3.6 on Complexity)
Theorem 3.6 means that the optimal value function is constant on $[s]_h$, the optimal Q-value function is constant on $[(s,a)]_h$, and an optimal policy exists which is also constant on $[(s,a)]_h$. The theorem additionally holds for any lifted policy which means that also non-optimal lifted policies induce value functions which respect the equivalence relation. This means that it is sufficient to operate on the reduced model to obtain good or optimal policies of the original model without any loss of information through the state and state-action space compression. Because the complexity depends heavily on the size of the state and state-action spaces (see Section 2.2.3) the reduced models can be solved faster.
The minimum savings are a reduction in the number of stored values (less computer memory needed) for any policy and value function proportional to $\frac{|\mathcal{S}_2|}{|\mathcal{S}_1|}$ or $\frac{|\mathcal{SA}_2|}{|\mathcal{SA}_1|}$, respectively. A more important reduction applies to the computational time: because the complexity formulae are worse than linear for a non-sparse transition function the savings are, in this case, proportional to the corresponding powers and products of $|\mathcal{S}|$ and $|\mathcal{SA}|$. Nevertheless, e. g. the number of value iterations to achieve a given numerical precision ε does not change – only for asynchronous updates as Gauss-Seidel or RL techniques could this happen. This represents the fact that a reducable model inherits some of the basic "difficultiness" of the reduced model although the theoretical complexity increases for larger models.

3.1.2 Symmetries by Group Actions on MDPs

In this section the framework of MDP homomorphisms is to be related to the classical notion of symmetry by means of group actions which turns out to be similar but not exactly the same.[11] [170] gives an example for which the two concepts of symmetry are different. However, the concept of MDP homomorphisms includes the symmetry group concept which is defined by group actions of the group of all MDP automorphisms.[12] [70] also uses this approach (called *bisimulation* there) and relates it to *finite state machines* (FSM). One complexity result of [70] is that it is NP-hard to determine whether a given finite model is already minimal under reduction by symmetry groups.

Firstly, some standard definitions of group homomorphisms and group actions are recalled (for more details see Appendix A). A (left) group action Θ is a map $\Theta : G \times X \to X$ such that for all $g, h \in G$ and for all $x \in X$ holds: $\Theta(g, \Theta(h, x)) = \Theta(gh, x)$ and $\Theta(1, x) = x$ whereas $1 \in G$ is the identity. A simplified notation for group actions is $gh \cdot x = ghx = \Theta(g, \Theta(h, x))$. Furthermore, group actions act like permutations on X. This can be made precise by introducing the kernel of a group action. The main results related to equivalence classes are Proposition A.4 and Proposition A.5 in which it is shown that equivalence relations and group actions are equivalent concepts.

Proposition A.5 is now to be applied to MDPs. Therefore, a definition of symmetries based on MDP automorphisms is given:

3.8 Definition (MDP Iso- and Automorphism, Symmetry Group [170])
Let $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SA}_1, T_1, R_1)$ *and* $\mathcal{M}_2 = (\mathcal{D}_2, \mathcal{S}_2, \mathcal{SA}_2, T_2, R_2)$ *be two MDPs and let* $h : \mathcal{SA}_1 \to \mathcal{SA}_2$ *be an MDP homomorphism.* h *is said to be an MDP isomorphism if it is bijective*[13] *and it is called an MDP automorphism if it is bijective and if* $\mathcal{SA}_1 = \mathcal{SA}_2$. *The set of all automorphisms* $\mathrm{Aut}(\mathcal{M}_1)$ *of an MDP* $\mathcal{M}_1$ *is called the symmetry group of the MDP.*[14]

3.9 Corollary (Symmetry Group of an MDP and MDP homomorphisms)
Let $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SA}_1, T_1, R_1)$ be an MDP and let $G = \mathrm{Aut}(\mathcal{M}_1)$ be its symmetry group. Then, $\Theta : G \times \mathcal{SA}_1 \to \mathcal{SA}_1$, $\Theta(g, (s, a)) = g(s, a)$, is a group action and there exists an MDP homomorphism h which induces the same equivalence relation as the group action.

Proof: For the proof previous results have to be collected. Θ is a group action because $\mathrm{id}_{\mathcal{SA}_1} \in G$ is the neutral element of $(G, \circ)$, and the neutral element and the composition of automorphisms fulfill the group action axioms. Then, an equivalence relation on $\mathcal{SA}_1$ with the group orbits as equivalence classes is induced. To be able to define an MDP symmetry it is to show that Equations 3.2, 3.3, 3.8, and 3.7 hold for the equivalence relation defined by the G-orbits.[15]

[11] A corresponding citation from the Wikipedia website on equivalence relations motivates to complete our description (`http://en.wikipedia.org/wiki/Equivalence_relation`, 06.09.2007): "It is very well known that lattice theory captures the mathematical structure of order relations. It is much less known that transformation groups (some authors prefer permutation groups) and their orbits capture the mathematical structure of equivalence relations." This web page includes also important suggestions for literature presented below.

[12] In principle, the main results of this section about MDPs can be found in [170] but there are no pointers to the relation to group actions.

[13] $h(s, a) = (f(s), g_s(a))$ is bijective iff f, g_s are all bijective.

[14] In fact, the group properties hold for $(\mathrm{Aut}(\mathcal{M}_1), \circ)$.

[15] This does not directly follow due to the fact that G consists of MDP automorphisms. For the corresponding MDP homomorphism some conditions become nearly trivial since the induced equivalence classes of MDP automorphisms are singletons.

Equations 3.2 and 3.3 follow from 3.1: Let $(s,a''),(s,a) \in \mathcal{SA}_1$ and let $s' \in \Pi_s([(s,a'')])$, i. e. there exists an MDP automorphism $h \in G$ with $h(s,a'') = (f(s), g_s(a'')) = (s', g_s(a''))$ which especially means $f(s) = s'$. Then $h(s,a) = (f(s), g_s(a)) = (s', g_s(a))$ and it follows with $a' = g_s(a)$ that $\exists a' \in \mathcal{A}_1(s') : \ (s',a') \in [(s,a)]$ because (s,a) and (s',a') are in the same G-orbit.
Equation 3.8 follows because for any automorphism $h \in G$ and any $(s',a') = h(s,a)$ the reward $R_1(s,a) = R_1(h(s,a))$. For Equation 3.7 to prove note that for an automorphism $h(s,a) = (f(s), g_s(a))$ Equation 3.4 turns into $T_1(s,a,s') = T_1(h(s,a), f(s'))$. Furthermore, $G_s = \Pi_s G = \{f : \ h \in G \text{ with } h(s,a) = (f(s), g_s(a))\}$ is a group because G is a group, G_s induces a group action on $\mathcal{S}_1$ by function evaluation and composition (as G on $\mathcal{SA}_1$), and the orbits of G_s are by definition the projections of orbits of G, i. e. the equivalence classes on $\mathcal{S}_1$. Then, for any $h \in G$ and any $(s',a') = h(s,a) = (f(s), g_s(a))$ holds $(f \circ G_s) \cdot s = G_s \cdot s$ (since $f \in G_s$), and

$$\begin{aligned}
\widetilde{T}_1(s,a,[s'']) &= \sum_{s''' \in G_s \cdot s''} T_1(s,a,s''') \\
&= \sum_{s''' \in G_s \cdot s''} T_1(h(s,a), f(s''')) \\
&= \sum_{s''' \in G_s \cdot s''} T_1(h(s,a), s''') \\
&= \widetilde{T}_1(s',a',[s'']) \,.
\end{aligned}$$

Summarised, the equivalence relation on $\mathcal{SA}_1$ induces one on $\mathcal{S}_1$ and a corresponding MDP symmetry. Finally, Lemma 3.3 shows the existence of an MDP homomorphism which induces the same equivalence relation. □

Although MDP homomorphisms can capture equivalence relations induced by (subgroups of) the symmetry group the reverse implication is not true because for MDP isomorphisms Equation 3.4 reduces to

$$\forall s \in \mathcal{S}_1 \ \forall s' \in \mathcal{S}_1 \ \forall a \in \mathcal{A}(s) : T_1(s,a,s') = T_2(f(s), g_s(a), f(s')) \,.$$

[170] contains an example of model reductions due to MDP homomorphisms[(16)] which can not be obtained by symmetry groups of MDPs.

3.2 Homomorphisms and Symmetry in 2P-ZS-MGs

In Section 3.1 the concepts of MDP homomorphisms and MDP symmetries have been shown to be equivalent. This reduces the burden to generalise both concepts to 2P-ZS-MGs to the task to choose one of them. In the following, the concept of MDP homomorphisms is generalised to 2P-ZS-MG homomorphisms because that formalism explicitly includes the mappings from a model to the reduced model in its definition. Nevertheless, the formalism of 2P-ZS-MG homomorphisms could be transformed to one of 2P-ZS-MG symmetries.
The main result of Section 3.2.1 is Theorem 3.16 which includes Theorem 3.6 (MDPs) and states that a symmetric (or reducable) 2P-ZS-MG induces a structure on the optimal value functions that respects this symmetry (or reduction) and that a corresponding policy exists. The proof of the main theorem was independently given by the author and goes

[(16)] Characterised by reward respecting SSP partitions, in fact equalling MDP symmetries.

beyond the proof for MDPs e. g. because the matrix game reduction (MGR) property (Definition 3.12) needs to be valid which is always true (Proposition 3.13). It is to be noted that (asynchronous) RL algorithms operating on the original model can behave differently than that operating on the reduced model because for the latter the experience is spread over all equivalent states even if never visited by the learning agent.

Besides the symmetries of MDPs, a qualitatively new symmetry that results from exchanging the two agents can be exploited. For that purpose, the concept of μ-homomorphisms is introduced in Section 3.2.2 which can also be utilised to identify identical models which only differ by a scaling of the rewards. This can not be achieved by the standard 2P-ZS-MG homomorphism framework. Practically, agent exchanging symmetries have been used for different board games by the argument that exchanging the agents in a zero-sum game has to result in a multiplication of the value by -1. One example is given by the pioneer Samuel for checkers [179].[(17)] The present work lays formal foundations for this practice and, more importantly, shows that the exchange of agents is also compatible with other standard symmetries similar to that of 2P-ZS-MGs (Proposition 3.24).

3.2.1 2P-ZS-MG Homomorphisms and Symmetry

In this section the analogue concepts of Section 3.1 for 2P-ZS-MGs are to be introduced. The main differences are that the state-action spaces $\mathcal{SA}_i$ have to be replaced by the corresponding state-action spaces $\mathcal{SAO}_i$ and that additional projection properties hold. This implicates that for the projection Π_s and the reward and transition functions some changes occur.

First and foremost, again some aspects of projecting partitions of the state-action space onto partitions of the state space have to be noted. According to Section 3.1, a first assumption is that a partition of the (finite) state-action space $\mathcal{P}(\mathcal{SAO}_1) = \{\,[(s,a,o)] : (s,a,o) \in \mathcal{SAO}_1\}$ into equivalence classes is given. It is further assumed that this partition induces a partition of equivalence classes on the state space $\mathcal{P}(\mathcal{S}_1) = \{\,[s] : s \in \mathcal{S}_1\}$ by the direct *projection* $\Pi_s : \mathcal{SAO}_1 \to \mathcal{S}_1$ with $\Pi_s(s_1,a_1,o_1) = s_1$ by means of $[s] = \Pi_s([(s,a,o)])$. The well-definedness of this projection onto state equivalence classes is equivalent to the condition

$$\forall (s,a'',o'') \in \mathcal{SAO}_1\ \forall (s,a,o) \in \mathcal{SAO}_1\ \forall s' \in \Pi_s([(s,a'',o'')]) \\ \exists (a',o') \in \mathcal{A}_1(s') \times \mathcal{O}_1(s') :\ (s',a',o') \in [(s,a,o)]\,. \quad (3.10)$$

If the above equation is fulfilled then it can be rewritten in terms of the induced equivalence classes because $[s] = \Pi_s([(s,a,o)]) = \Pi_s([(s,a'',o'')])$:

$$\forall (s,a,o) \in \mathcal{SAO}_1\ \forall s' \in [s]\ \exists (a',o') \in \mathcal{A}_1(s') \times \mathcal{O}_1(s') :\ (s',a',o') \in [(s,a,o)]\,. \quad (3.11)$$

The well-definedness of the projection of equivalence classes by Π_s further implies the following condition on the equivalence classes:

$$\forall (s,a,o) \in \mathcal{SAO}_1\ \forall (s',a',o') \in \mathcal{SA}_1 :\ \Big((s',a',o') \in [(s,a,o)] \Rightarrow s' \in [s] \Big)\,. \quad (3.12)$$

Considering the reverse direction, i. e. if equivalence classes are given on $\mathcal{S}_1$ and $\mathcal{SAO}_1$ which satisfy Equations 3.2 and 3.3 then the projection Π_s exactly maps equivalence classes from $\mathcal{SAO}_1$ to the ones of $\mathcal{S}_1$.

[(17)] Samuel states that all stored board positions are transformed as if Black has to move.

The definitions above are a straightforward adaption from the MDP case. However, it will be obvious later in the proof of the main theorem (Theorem 3.16) that it is necessary to use the matrix game reduction property (Definition 3.12) which on its part uses the policy lifting (Definition 3.11) in 2P-ZS-MGs. For a lifted policy to depend only on the actions of one agent it is necessary that the state-action equivalence classes for joint actions induce equivalence classes on $\mathcal{SA}_1$ and $\mathcal{SO}_1$ by the projections $\Pi_{s,a} : \mathcal{SAO}_1 \to \mathcal{SA}_1$ and $\Pi_{s,o} : \mathcal{SAO}_1 \to \mathcal{SO}_1$, respectively. This means that the following two variants of Equation 3.10 have also to be valid:

$$\forall (s,a,o'') \in \mathcal{SAO}_1 \; \forall (s,a,o) \in \mathcal{SAO}_1 \; \forall (s',a') \in \Pi_{s,a}([(s,a,o'')]) \\ \exists o' \in \mathcal{O}_1(s') : \; (s',a',o') \in [(s,a,o)] \quad (3.13)$$

and

$$\forall (s,a'',o) \in \mathcal{SAO}_1 \; \forall (s,a,o) \in \mathcal{SAO}_1 \; \forall (s',o') \in \Pi_{s,o}([(s,a'',o)]) \\ \exists a' \in \mathcal{A}_1(s') : \; (s',a',o') \in [(s,a,o)] \,. \quad (3.14)$$

If they are fulfilled, these two equations can also be abbreviated by the following:

$$\forall (s,a,o) \in \mathcal{SAO}_1 \; \forall (s',a') \in [s,a] \; \exists o' \in \mathcal{O}_1(s') : \; (s',a',o') \in [(s,a,o)] \quad (3.15)$$

and

$$\forall (s,a,o) \in \mathcal{SAO}_1 \; \forall (s',o') \in [s,o] \; \exists a' \in \mathcal{A}_1(s') : \; (s',a',o') \in [(s,a,o)] \,. \quad (3.16)$$

Next, 2P-ZS-MG homomorphisms can be defined which do *not* capture symmetries exchanging the two agents. The straightforward generalisation from MDP homomorphisms would be $h(s,a,o) = (f(s), g_s(a,o))$ but instead the homomorphism is defined as $h(s,a,o) = (f(s), g_s(a), i_s(o))$ because this explicitly takes the projective properties (Equations 3.11, 3.15 and 3.16) into account, whereas $[s] = f^{-1}(f(s))$, $[(s,a)] = \{(s',a') : g_{s'}(a') = g_s(a)\}$, and $[(s,o)] = \{(s',o') : i_{s'}(o') = i_s(o)\}$.

3.10 Definition (2P-ZS-MG Homomorphism)
Let $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SAO}_1, T_1, R_1)$ and $\mathcal{M}_2 = (\mathcal{D}_2, \mathcal{S}_2, \mathcal{SAO}_2, T_2, R_2)$ be two 2P-ZG-MGs as defined in Section 2.4 with $\mathcal{D}_1 = \mathcal{D}_2 = \mathbb{N}_0$. A map $h : \mathcal{SAO}_1 \to \mathcal{SAO}_2$ is called a 2P-ZS-MG homomorphism if h is a surjection, defined by a tuple of surjective maps $(f, (g_s)_{s\in\mathcal{S}_1}, (i_s)_{s\in\mathcal{S}_1})$ with $h(s,a,o) = (f(s), g_s(a), i_s(o))$, $f : \mathcal{S}_1 \to \mathcal{S}_2$, $g_s : \mathcal{A}_1(s) \to \mathcal{A}_2(f(s))$, and $i_s : \mathcal{O}_1(s) \to \mathcal{O}_2(f(s))$ such that

$$\forall (s,a,o) \in \mathcal{SAO}_1 \; \forall s' \in \mathcal{S}_1 : \; \widetilde{T}_1(s,a,o,[s']) = T_2(f(s), g_s(a), i_s(o), f(s')) \quad (3.17)$$

and

$$\forall (s,a,o) \in \mathcal{SAO}_1 : \; R_1(s,a,o) = R_2(f(s), g_s(a), i_s(o)) \quad (3.18)$$

where the block transition function $\widetilde{T}_1$ *of the 2P-ZS-MG $\mathcal{M}_1$ is defined by*

$$\widetilde{T}_1 : \mathcal{SAO}_1 \times \{[s] : s \in \mathcal{S}_1\} \to \mathbb{R}, \; \widetilde{T}_1(s,a,o,[s']) = \sum_{s''\in[s']} T_1(s,a,o,s'') \quad (3.19)$$

and the equivalence classes $[(s,a,o)]$ on $\mathcal{SA}_1$ are defined by $(s',a',o') \in [(s,a,o)]$ iff $h(s',a',o') = h(s,a,o)$ and the equivalence classes $[s]$ on $\mathcal{S}_1$, $[(s,a)]$ on $\mathcal{SA}_1$, and $[s,o]$ on $\mathcal{SO}_1$ are defined by the projections $\Pi_s([(s,a,o)])$, $\Pi_{s,a}([(s,a,o)])$, and $\Pi_{s,o}([(s,a,o)])$, respectively, which makes Equations 3.10, 3.13 and 3.14 necessary conditions.

According to MDPs, it is necessary to have the possibility to lift policies. This is only possible because the projections $\Pi_{s,a}$ and $\Pi_{s,o}$ project onto equivalence classes:

3.11 Definition (Policy Lifting for 2P-ZS-MGs)
Let $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SAO}_1, T_1, R_1)$ and $\mathcal{M}_2 = (\mathcal{D}_2, \mathcal{S}_2, \mathcal{SAO}_2, T_2, R_2)$ be two 2P-ZS-MGs and let $h : \mathcal{SAO}_1 \to \mathcal{SAO}_2$, $h(s,a,o) = (f(s), g_s(a), i_s(o))$ be an 2P-ZS-MG homomorphism. Let $\widehat{\pi}$ be a policy of the 2P-ZS-MG $\mathcal{M}_2$ for the first or second agent, appropriately. Then the corresponding lifted policy π of the 2P-ZS-MG $\mathcal{M}_1$ is defined for the first agent by

$$\pi(s,a) = \frac{\widehat{\pi}(f(s), g_s(a))}{|g_s^{-1}(g_s(a))|}\,, \tag{3.20}$$

and for the second agent by

$$\pi(s,o) = \frac{\widehat{\pi}(f(s), i_s(o))}{|i_s^{-1}(i_s(o))|}\,. \tag{3.21}$$

Next, the matrix game reduction property which will be essential for the reduction by 2P-ZS-MG homomorphisms is introduced:

3.12 Definition (Matrix Game Reduction (MGR) Property)
Let $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SAO}_1, T_1, R_1)$ and $\mathcal{M}_2 = (\mathcal{D}_2, \mathcal{S}_2, \mathcal{SAO}_2, T_2, R_2)$ be two 2P-ZS-MGs and let $h : \mathcal{SAO}_1 \to \mathcal{SAO}_2$, $h(s,a,o) = (f(s), g_s(a), i_s(o))$ be an 2P-ZS-MG homomorphism. Then h is said to have the matrix game reduction (MGR) property iff for all states $s \in \mathcal{S}_1$, for all matrices $M_{1,s} \in \mathbb{R}^{|\mathcal{A}_1(s)|, |\mathcal{O}_1(s)|}$ and $M_{2,f(s)} \in \mathbb{R}^{|\mathcal{A}_2(f(s))|, |\mathcal{O}_2(f(s))|}$ that respect the structure of h, i. e. $\forall (a', o') \in \mathcal{A}_2(f(s)) \times \mathcal{O}_2(f(s))\ \forall (a,o) \in g_s^{-1}(a') \times i_s^{-1}(o') : (M_{1,s})(a,o) = (M_{2,f(s)})(a',o')$, holds that

$$V^*(M_{1,s}) = V^*(M_{2,f(s)}) \tag{3.22}$$

and additionally that the optimal policy of $M_{1,s}$ is a lifted optimal policy of $M_{2,f(s)}$ (interpreting the matrix game as a single state 2P-ZS-MG as in Definition 2.19).

By definition Equation 3.22 is equivalent to[18]

$$\begin{aligned}\max_{\pi_s \in \mathrm{PD}(\mathcal{A}_1(s))} \min_{o \in \mathcal{O}_1(s)} \sum_{a \in \mathcal{A}_1(s)} \pi_s(a) \cdot M_{1,s}(a,o) \\ = \max_{\pi_{f(s)} \in \mathrm{PD}(\mathcal{A}_2(f(s)))} \min_{o \in \mathcal{O}_2(f(s))} \sum_{a \in \mathcal{A}_2(f(s))} \pi_{f(s)}(a) \cdot M_{2,f(s)}(a,o)\,,\end{aligned} \tag{3.23}$$

where the abbreviation $M(a,o) = M_{i,j}$ if $a_i = a$ and $o_j = o$ is used.

3.13 Proposition (Validity of the MGR Property)
Let $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SAO}_1, T_1, R_1)$ and $\mathcal{M}_2 = (\mathcal{D}_2, \mathcal{S}_2, \mathcal{SAO}_2, T_2, R_2)$ be two 2P-ZS-MGs and let $h : \mathcal{SAO}_1 \to \mathcal{SAO}_2$, $h(s,a,o) = (f(s), g_s(a), i_s(o))$ be a 2P-ZS-MG homomorphism. Then the MGR property holds.

[18] Because of the minimax theorem (Theorem 2.23), the definition of the value of a matrix game (Definition 2.21) and the fact that the minimum over probability distributions for the later deciding agent (applied first in the equation) can be replaced by the pure actions.

Proof: Let $\mathcal{M}_1$, $\mathcal{M}_2$, h, $M_{1,s}$, $M_{2,f(s)}$ be as in Definition 3.12. It will be shown that $M_{1,s}$ can be transformed into $M_{2,f(s)}$ by removing equal rows and equal columns, and hence the associated matrix games have the same value. Since the structure of equal rows and columns corresponds to the structure of the state-wise intersection of equivalence classes $[(s,a)] \cap (\{s\} \times \mathcal{A}_1(s)) = \{s\} \times g_s^{-1}(g_s(a))$ and $[(s,o)] \cap (\{s\} \times \mathcal{O}_1(s)) = i_s^{-1}(i_s(o))$, respectively, an optimal policy from $M_{2,f(s)}$ can be lifted to a policy of $M_{1,s}$.
For a fixed $s \in \mathcal{S}_1$ be $\{a'_1, \dots, a'_{m_2}\} = \mathcal{A}_2(f(s))$ an enumeration of the actions with $m_2 = |\mathcal{A}_2(f(s))|$, and be $\{o'_1, \dots, o'_{n_2}\} = \mathcal{O}_2(f(s))$ an enumeration of the actions with $n_2 = |\mathcal{O}_2(f(s))|$ such that the matrix is ordered as $M_{2,f(s)}(i,j) = M_{2,f(s)}(a'_i, o'_j)$. Since $g_s : \mathcal{A}_1(s) \to \mathcal{A}_2(f(s))$ is surjective, $g_s^{-1}(a'_i) \neq \emptyset$ for every i and there exists an enumeration $\{a_1, \dots, a_{m_1}\} = \mathcal{A}_1(s)$ with $a_k \in g_s^{-1}(a'_k)$ for $1 \leq k \leq m_2$. Since $\mathcal{A}_1(s) = \bigcup_{a' \in \mathcal{A}_2(f(s))} g_s^{-1}(a')$, for all $k > m_2$ holds that $a_k \in g_s^{-1}(g_s(a_{i_k}))$ with $i_k \leq m_2$. Then for all $o \in \mathcal{O}_1(s)$ holds: $h(s, a_k, o) = (f(s), g_s(a_k), i_s(o)) = (f(s), g_s(a_{i_k}), i_s(o)) = h(s, a_{i_k}, o)$. This means that in $M_{1,s}$ for each $k > m_2$ the k-th row is equal to the i_k-th row where $i_k \leq m_2$ and therefore all rows with index $k \geq m_2$ can be removed. Further, all actions $g_s^{-1}(g_s(a_{i_k})) \subseteq \mathcal{A}_1(s)$ are equal which shows that the policy for agent P_1 can be lifted.
For columns the analogous statement holds that there exists an enumeration $\{o_1, \dots, o_{n_1}\} = \mathcal{O}_1(s)$ with $o_k \in i_s^{-1}(o'_k)$ for $1 \leq k \leq n_2$ and that all columns with index $k > n_2$ can be removed. Further, all actions $i_s^{-1}(i_s(o_{i_k})) \subseteq \mathcal{O}_1(s)$ are equal which shows that the policy for agent P_2 can also be lifted. Since by removing equal rows equal columns stay equal, $M_{1,s}$ can be transformed into $\overline{M}_{1,s}$ which equals by construction $M_{2,f(s)}$.

□

3.14 Remark (MGR Property for MDPs)
For MDPs, the MGR property can be interpretated by M_1 being a vector (the minimiser has only one action to "choose") which makes the proof of Proposition 3.13 as simple as $\max_{a \in \mathcal{A}_1(s)} M_{1,s}(a) = \max_{a \in \mathcal{A}_2(f(s))} M_{2,f(s)}(a)$ by noting that $g_s : \mathcal{A}_1(s) \to \mathcal{A}_2(f(s))$ is surjective.[19]

3.15 Example (Structure of Matrices with MGR Property)
The definition of the MGR property is in principle independent from a state s although it must hold for all states. Therefore, only two matrices M_1 and M_2 are presented which can be thought of as $M_{1,s}(a,o)$ and $M_{2,f(s)}(a',o')$:

$$M_1 = \left(\begin{array}{cc|c} c_1 & c_1 & c_2 \\ c_1 & c_1 & c_2 \\ c_1 & c_1 & c_2 \\ \hline c_3 & c_3 & c_4 \end{array}\right), \quad M_2 = \left(\begin{array}{c|c} c_1 & c_2 \\ \hline c_3 & c_4 \end{array}\right).$$

The lines indicate the borders of different equivalent action pairs. There has to exist an enumeration of the actions such that the lines generate a grid on the matrix to fulfill the MGR property. The type of the grid is defined by the equivalence classes of a 2P-ZS-MG while the numbers $c_i \in \mathbb{R}$ are arbitrary.

3.16 Theorem (Main Implications of 2P-ZS-MG Homomorphisms)
Let $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SAO}_1, T_1, R_1)$ be a 2P-ZS-MG and let $h : \mathcal{SAO}_1 \to \mathcal{SAO}_2$ be an 2P-ZS-MG homomorphism (Definition 3.10) for some 2P-ZS-MG $\mathcal{M}_2 = (\mathcal{D}_2, \mathcal{S}_2, \mathcal{SAO}_2, T_2, R_2)$.

[19] The simplicity of this argument could be the reason why it is not mentioned in [170]. However, for matrix games the sitation is a little less simple.

Then, $V^*(s) = V^*(f(s))$ and $Q^*(s,a,o) = Q^*(h(s,a,o))$. Furthermore, there exists an optimal policy of $\mathcal{M}_1$ which is a lifted optimal policy of $\mathcal{M}_2$.

Proof: Parts of the proof are similar to that of Theorem 3.6 in [170][(20)] and [217][(21)] but independently developed and the theorem here is valid for a larger class of problems. Let V_k be the k-th value iterate for $V_0 = 0$ (it is only necessary that V_0 respects the symmetry but this one respects all possible symmetries), i. e. $V_k = V_k(Q_{k-1})$ according to Equation 2.48, and $Q_k = Q_k(V_k)$ according to Equation 2.49. It will be shown by induction that each of the (Q-)value iterates keeps the same symmetry which implies that the limits V^* and Q^*, respectively also possess this symmetry.
Induction starts by noticing that $V_0 = 0$ respects any symmetry and that then $Q_0 = R_1$ also respects any symmetry which can be induced by h by its definition. The induction hypothesis (I. H.) is the assumption that this is true for all iterates up to V_{k-1}, Q_{k-1}, i. e. for all $j \leq k-1$ and for $h(s,a,o) = (f(s), g_s(a), i_s(o))$ holds that $V_j(s) = V_j(f(s))$ and $Q_j(s,a,o) = Q_j(h(s,a,o))$. Here, the value functions are not indexed like the state spaces because the argument uniquely shows which value function is meant. Then, for $h(s,a,o) = (f(s), g_s(a), i_s(o))$ holds:

$$\begin{aligned}
V_k(s) &= \max_{\pi_s \in \mathrm{PD}(\mathcal{A}_1(s))} \min_{o \in \mathcal{O}_1(s)} \sum_{a \in \mathcal{A}_1(s)} \pi_s(a) \cdot Q_{k-1}(s,a,o) \\
&\overset{\text{I. H.}}{=} \max_{\pi_s \in \mathrm{PD}(\mathcal{A}_1(s))} \min_{o \in \mathcal{O}_1(s)} \sum_{a \in \mathcal{A}_1(s)} \pi_s(a) \cdot Q_{k-1}(f(s), g_s(a), i_s(o)) \\
&\overset{(1)}{=} \max_{\pi_{f(s)} \in \mathrm{PD}(\mathcal{A}_2(f(s)))} \min_{o \in \mathcal{O}_2(f(s))} \sum_{a \in \mathcal{A}_2(f(s))} \pi_{f(s)}(a) \cdot Q_{k-1}(f(s), a, o) \\
&= V_k(f(s)),
\end{aligned}$$

whereas (1) is the MGR property (Definition 3.12 and below) which is always fulfilled (Proposition 3.13) and also guarantees the existence of a lifted policy for every state s. Furthermore,

$$\begin{aligned}
Q_k(s,a,o) &= R_1(s,a,o) + \gamma \sum_{s' \in \mathcal{S}_1} T_1(s,a,o,s') \cdot V_k(s') \\
&\overset{(1)}{=} R_2(h(s,a,o)) + \gamma \sum_{s' \in \mathcal{S}_1} T_1(s,a,o,s') \cdot V_k(f(s')) \\
&\overset{(2)}{=} R_2(h(s,a,o)) + \gamma \sum_{f(s') \in \mathcal{S}_2} V_k(f(s')) \sum_{s'' \in f^{-1}(f(s'))} T_1(s,a,o,s'') \\
&\overset{(3)}{=} R_2(h(s,a,o)) + \gamma \sum_{f(s') \in \mathcal{S}_2} V_k(f(s')) \cdot \widetilde{T}_1(s,a,o,[s']) \\
&\overset{(4)}{=} R_2(h(s,a,o)) + \gamma \sum_{f(s') \in \mathcal{S}_2} T_2(h(s,a,o), f(s')) \cdot V_k(f(s')) \\
&\overset{(5)}{=} R_2(h(s,a,o)) + \gamma \sum_{s' \in \mathcal{S}_2} T_2(h(s,a,o), s') \cdot V_k(s') \\
&= Q_k(h(s,a,o)).
\end{aligned}$$

[(20)] The author thinks that for the second equality sign of the proof in [170] some implicit assumptions on the state-value function V_m are made which are more obvious in the presentation here.
[(21)] That proof on the other hand is more formal but possibly more complex than necessary.

(1) is valid because of the definition of h and because the symmetry of V_k was shown directly before, for (2) note that $f : \mathcal{S}_1 \to \mathcal{S}_2$, (3) holds by definition of the block transition function $\widetilde{T}_1$ and because $f^{-1}(f(s')) = [s']$ and (4) again by definition, and (5) because f is surjective. □

3.17 Remark (Implications of Theorem 3.16 on Complexity)
Remark 3.7 is analogously valid for 2P-ZS-MGs. The optimal value function is especially constant on $[s]_h$, the optimal Q-value function is constant on $[(s,a,o)]_h$, and there exists an optimal policy for each agent which is constant on $[(s,a)]_h$ and $[(s,o)]_h$, respectively.

3.2.2 Automorphisms for the Exchange of Agents

After the analogue symmetries of MDPs in 2P-ZS-MGs have been exploited, a qualitatively new symmetry that results from exchanging the two agents is introduced.[(22)] For that purpose, the concept of μ-homomorphisms is employed. Practically, agent exchanging symmetries have been used for different board games – as already mentioned in the introduction of this chapter – but the present work lays formal foundations for this practice and, more importantly, shows that the exchange of agents is also compatible with the other standard symmetries obtained by 2P-ZS-MG homomorphisms (Proposition 3.24).

Briefly summarised, a 2P-ZS-MG μ-homomorphism is a 2P-ZS-MG homomorphism for which the reward condition is changed by a factor of μ. If $\mu > 0$ then the proof of Theorem 3.16 holds with the simple changes that $R_1(s,a,o) = \mu \cdot R_2(h(s,a,o))$, $V_k(s) = \mu \cdot V_k(f(s))$, and $Q_k(s) = \mu \cdot Q_k(f(s))$ because a *positive* constant can be factored out of an expression to maximise or to minimise: $\max_x(\mu \cdot f(x)) = \mu \cdot \max_x(f(x))$.
However, for a *negative* constant $\mu < 0$ a maximums turns into a minimum and vice versa: $\max_x(\mu \cdot f(x)) = \mu \cdot \min_x(f(x))$. This makes the two cases essentially different and justifies the separate definition. In fact, the case $\mu > 0$ only introduces an additional scaling of the reward to the framework of Section 3.2.1 but $\mu < 0$ points to the new aspect of agent exchanging symmetries in 2P-ZS-MGs.

Now, 2P-ZS-MG μ-homomorphisms are to be defined:

3.18 Definition (2P-ZS-MG μ-Homomorphism)
Let $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SAO}_1, T_1, R_1)$ and $\mathcal{M}_2 = (\mathcal{D}_2, \mathcal{S}_2, \mathcal{SAO}_2, T_2, R_2)$ be two 2P-ZG-MGs as defined in Section 2.4 with $\mathcal{D}_1 = \mathcal{D}_2 = \mathbb{N}_0$. A map $h : \mathcal{SAO}_1 \to \mathcal{SAO}_2$ is called an 2P-ZS-MG μ-homomorphism if $\mu \neq 0$, h is a surjection, and additionally holds:

1. *if $\mu > 0$: h is defined by a tuple of surjective maps $(f, (g_s)_{s\in\mathcal{S}_1}, (i_s)_{s\in\mathcal{S}_1})$ with $h(s,a,o) = (f(s), g_s(a), i_s(o))$, $f : \mathcal{S}_1 \to \mathcal{S}_2$, $g_s : \mathcal{A}_1(s) \to \mathcal{A}_2(f(s))$, and $i_s : \mathcal{O}_1(s) \to \mathcal{O}_2(f(s))$,*
2. *if $\mu < 0$: h is defined by a tuple of surjective maps $(f, (g_s)_{s\in\mathcal{S}_1}, (i_s)_{s\in\mathcal{S}_1})$ with $h(s,a,o) = (f(s), i_s(o), g_s(a))$, $f : \mathcal{S}_1 \to \mathcal{S}_2$, $g_s : \mathcal{A}_1(s) \to \mathcal{O}_2(f(s))$, and $i_s : \mathcal{O}_1(s) \to \mathcal{A}_2(f(s))$,*

such that

$$\forall (s,a,o) \in \mathcal{SAO}_1 \ \forall s' \in \mathcal{S}_1 : \ \widetilde{T}_1(s,a,o,[s']) = T_2(h(s,a,o), f(s')) \tag{3.24}$$

and

$$\forall (s,a,o) \in \mathcal{SAO}_1 : \ R_1(s,a,o) = \mu \cdot R_2(h(s,a,o)) \tag{3.25}$$

where the block transition function $\widetilde{T}_1$ *of the 2P-ZS-MG $\mathcal{M}_1$ is defined as in Equation 3.19 and the equivalence classes fulfill the projective conditions of 2P-ZS-MG homomorphisms, which makes Equations 3.10, 3.13 and 3.14 necessary conditions.*

[(22)] This exchange of agents is not to be confused with a permutation of agents in a multi-player MDP.

The MGR property remains for $\mu > 0$ but significantly changes for $\mu < 0$:

3.19 Remark (Adaptation of MGR Property to μ-homomorphisms)
For $\mu < 0$, h is said to have the matrix game reduction (MGR) property iff for all states $s \in \mathcal{S}_1$, for all matrices $M_{1,s} \in \mathbb{R}^{|\mathcal{A}_1(s)|,|\mathcal{O}_1(s)|}$ and $M_{2,f(s)} \in \mathbb{R}^{|\mathcal{A}_2(f(s))|,|\mathcal{O}_2(f(s))|}$ that respect the structure of h, i. e. $\forall (a',o') \in \mathcal{A}_2(f(s)) \times \mathcal{O}_2(f(s))\ \forall (a,o) \in g_s^{-1}(o') \times i_s^{-1}(a') : (M_{1,s})(a,o) = -((M_{2,f(s)})(a',o'))^T$, holds that

$$V^*(M_{1,s}) = -V^*(M_{2,f(s)}) \tag{3.26}$$

and that the optimal policies of $M_{1,s}$ are lifted optimal policies of $M_{2,f(s)}$.
The proof that this property holds for every 2P-ZS-MG homomorphism is accordingly to that of Proposition 3.13 with the difference that it is to show that $M_{1,s}$ can be transformed into $\overline{M}_{1,s} = -M_{2,f(s)}^T$ by removing equal rows and equal columns. Then, it follows by the Minimax-Theorem (Theorem 2.23) which for any matrix N holds:

$$\max_{\pi_1}\min_{\pi_2} \pi_1^T N \pi_2 = \min_{\pi_2}\max_{\pi_1} \pi_1^T N \pi_2 = -\max_{\pi_2}\min_{\pi_1} -\pi_1^T N \pi_2 = -\max_{\pi_2}\min_{\pi_1} \pi_2^T(-N^T)\pi_1\,.$$

This means that $V^*(N) = -V^*(-N^T)$ and that a policy pair (π_1^*, π_2^*) in N is optimal iff (π_2^*, π_1^*) is optimal for in $-N^T$.
To see that $M_{1,s}$ can be transformed into $\overline{M}_{1,s} = -M_{2,f(s)}^T$ the following adaptations have to be made: the statement remains that the structure of equal rows and columns corresponds to the structure of the state-wise intersection of equivalence classes $[(s,a)] \cap (\{s\} \times \mathcal{A}_1(s)) = \{s\} \times g_s^{-1}(g_s(a))$ and $[(s,o)] \cap (\{s\} \times \mathcal{O}_1(s)) = i_s^{-1}(i_s(o))$, respectively, an optimal policy from $M_{2,f(s)}$ can be lifted to a policy of $M_{1,s}$.
Then some minor changes are necessary because now $g_s : \mathcal{A}_1(s) \to \mathcal{O}_2(f(s))$ and $i_s : \mathcal{A}_1(s) \to \mathcal{O}_2(f(s))$: For a fixed $s \in \mathcal{S}_1$ be $\{o'_1, \ldots, o'_{m_2}\} = \mathcal{O}_2(f(s))$ an enumeration of the actions with $m_2 = |\mathcal{O}_2(f(s))|$, and be $\{a'_1, \ldots, a'_{n_2}\} = \mathcal{A}_2(f(s))$ an enumeration of the actions with $n_2 = |\mathcal{A}_2(f(s))|$ such that the transposed matrix $(M_{2,f(s)}(i,j))^T$ is ordered as $(M_{2,f(s)}(i,j))^T = (M_{2,f(s)}(a'_i, o'_j))^T$. Since $g_s : \mathcal{A}_1(s) \to \mathcal{O}_2(f(s))$ is surjective, $g_s^{-1}(o'_i) \neq \emptyset$ for every i and there exists an enumeration $\{a_1, \ldots, a_{m_1}\} = \mathcal{A}_1(s)$ with $a_k \in g_s^{-1}(o'_k)$ for $1 \leq k \leq m_2$. Since $\mathcal{A}_1(s) = \bigcup_{o' \in \mathcal{O}_2(f(s))} g_s^{-1}(o')$, for all $k > m_2$ holds that $a_k \in g_s^{-1}(g_s(a_{i_k}))$ with $i_k \leq m_2$. Then for all $o \in \mathcal{O}_1(s)$ holds: $h(s,a_k,o) = (f(s), g_s(a_k), i_s(o)) = (f(s), g_s(a_{i_k}), i_s(o)) = h(s, a_{i_k}, o)$. This means that in $M_{1,s}$ for each $k > m_2$ the k-th row is equal to the i_k-th row where $i_k \leq m_2$ and therefore all rows with index $k > m_2$ can be removed. Further, all actions $g_s^{-1}(g_s(a_{i_k})) \subseteq \mathcal{A}_1(s)$ are equal which shows that the policy for P_1 can be lifted.
For columns the analogous statement holds that there exists an enumeration $\{o_1, \ldots, o_{n_1}\} = \mathcal{O}_1(s)$ with $o_k \in i_s^{-1}(a'_k)$ for $1 \leq k \leq n_2$ and that all columns with index $k > n_2$ can be removed. Further, all actions $i_s^{-1}(i_s(o_{i_k})) \subseteq \mathcal{O}_1(s)$ are equal which shows that the policy for P_2 can also be lifted. Since by removing equal rows equal columns stay equal, $M_{1,s}$ can be transformed into $\overline{M}_{1,s}$ which equals by construction $-M_{2,f(s)}^T$.

By definition Equation 3.26 is equivalent to

$$\begin{aligned}\max_{\pi_s \in \mathrm{PD}(\mathcal{A}_1(s))} \min_{o \in \mathcal{O}_1(s)} &\sum_{a \in \mathcal{A}_1(s)} \pi_s(a) \cdot M_{1,s}(a,o) \\ &= -\max_{\pi_{f(s)} \in \mathrm{PD}(\mathcal{A}_2(f(s)))} \min_{o \in \mathcal{O}_2(f(s))} \sum_{o \in \mathcal{O}_2(f(s))} \pi_{f(s)}(a) \cdot M_{2,f(s)}(a,o)\end{aligned} \tag{3.27}$$

which means

$$\max_{\pi_s \in \mathrm{PD}(\mathcal{A}_1(s))} \min_{o \in \mathcal{O}_1(s)} - \sum_{a \in \mathcal{A}_1(s)} \pi_s(a) \cdot (M_{2,f(s)}(g_s(a), i_s(o)))^T$$
$$= - \max_{\pi_{f(s)} \in \mathrm{PD}(\mathcal{A}_2(f(s)))} \min_{o \in \mathcal{O}_2(f(s))} \sum_{o \in \mathcal{O}_2(f(s))} \pi_{f(s)}(a) \cdot M_{2,f(s)}(a, o)\,. \quad (3.28)$$

In the following theorem all other corresponding main theorems of this chapter are included because 2P-ZS-MG homomorphisms are 2P-ZS-MG μ-homomorphisms with $\mu = 1 > 0$.

3.20 Theorem (Main Implications of 2P-ZS-MG μ-Homomorphisms)
Let $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SAO}_1, T_1, R_1)$ be a 2P-ZS-MG and let $h : \mathcal{SAO}_1 \to \mathcal{SAO}_2$ be an 2P-ZS-MG μ-homomorphism (Definition 3.18) for some 2P-ZS-MG $\mathcal{M}_2 = (\mathcal{D}_2, \mathcal{S}_2, \mathcal{SAO}_2, T_2, R_2)$. Then, $V^*(s) = \mu \cdot V^*(f(s))$ and $Q^*(s, a, o) = \mu \cdot Q^*(h(s, a, o))$. Furthermore, there exists an optimal policy of $\mathcal{M}_1$ which is a lifted optimal policy of $\mathcal{M}_2$ where in the case of $\mu < 0$ the two policies $\pi, \widehat{\pi}$ in each of the two formulae in Definition 3.11 are from different agents as the domains and co-domains of g_s and i_s indicate.

Proof: As mentioned at the beginning of this subsection, the case $\mu > 0$ just introduces a factor μ in the proof of Theorem 3.16 such that $R_1(s, a, o) = \mu \cdot R_2(h(s, a, o))$, $V_k(s) = \mu \cdot V_k(f(s))$, and $Q_k(s) = \mu \cdot Q_k(f(s))$.
However, for $\mu < 0$ something essentially different can be observed. In this case, the induction again starts by noticing that $V_0 = 0$ respects any symmetry and that then $Q_0 = R_1$ also respects any symmetry which can be induced by h. As induction hypothesis (I. H.), it is assumed that this is true for all iterates up to V_{k-1}, Q_{k-1}, i. e. for all $j \leq k-1$ and for $h(s, a, o) = (f(s), g_s(a), i_s(o))$ holds that $V_j(s) = \mu \cdot V_j(f(s))$ and $Q_j(s, a, o) = \mu \cdot Q_j(h(s, a, o))$. Again, the value functions are not indexed as the state spaces because the argument uniquely shows which value function is meant. Thus, for $h(s, a, o) = (f(s), i_s(o), g_s(a))$ holds:

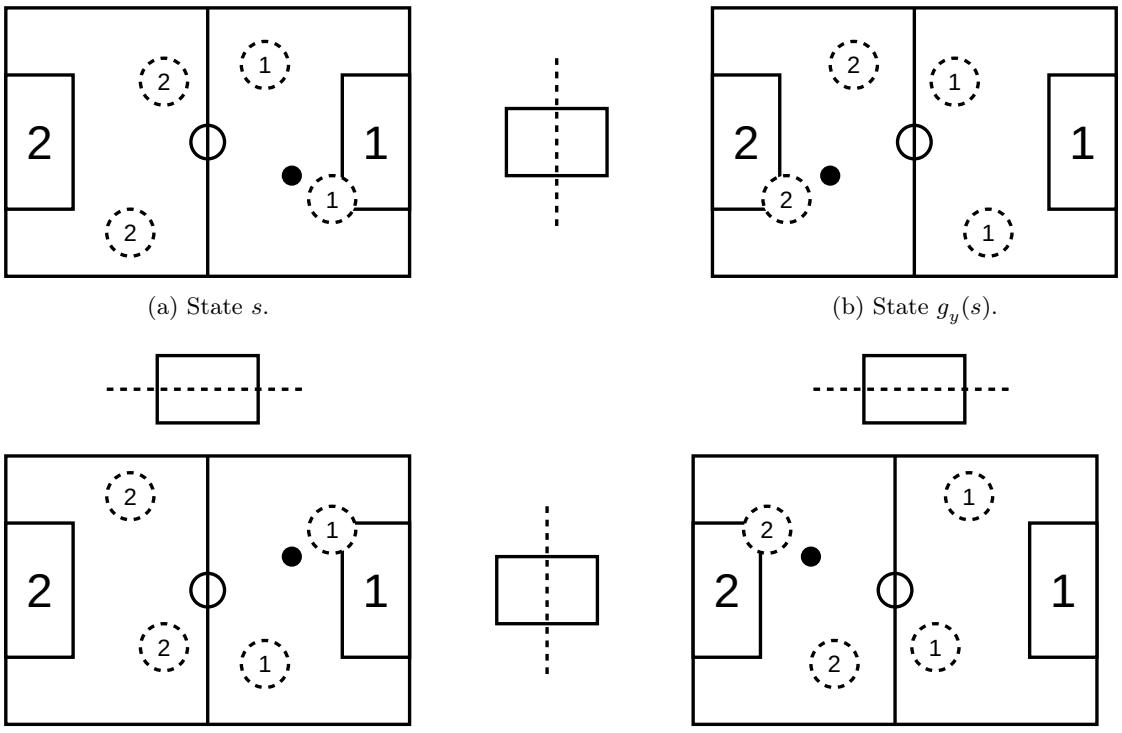
$$\begin{aligned}
V_k(s) &= \max_{\pi_s \in \mathrm{PD}(\mathcal{A}_1(s))} \min_{o \in \mathcal{O}_1(s)} \sum_{a \in \mathcal{A}_1(s)} \pi_s(a) \cdot Q_{k-1}(s, a, o) \\
&\overset{\text{I.H.}}{=} \max_{\pi_s \in \mathrm{PD}(\mathcal{A}_1(s))} \min_{o \in \mathcal{O}_1(s)} \sum_{a \in \mathcal{A}_1(s)} \pi_s(a) \cdot \mu \cdot Q_{k-1}(f(s), i_s(o), g_s(a)) \\
&\overset{(1)}{=} -\mu \cdot \max_{\pi_s \in \mathrm{PD}(\mathcal{A}_1(s))} \min_{o \in \mathcal{O}_1(s)} - \sum_{a \in \mathcal{A}_1(s)} \pi_s(a) \cdot Q_{k-1}(f(s), i_s(o), g_s(a)) \\
&\overset{(2)}{=} \mu \cdot \max_{\pi_{f(s)} \in \mathrm{PD}(\mathcal{A}_2(f(s)))} \min_{o \in \mathcal{O}_2(f(s))} \sum_{a \in \mathcal{A}_2(f(s))} \pi_{f(s)}(a) \cdot Q_{k-1}(f(s), a, o) \\
&= \mu \cdot V_k(f(s))\,,
\end{aligned}$$

whereas (1) holds because $-\mu > 0$ and (2) is the MGR property (Remark 3.19 and Equation 3.28) the proof of which needed the minimax theorem for matrix games (Theorem 2.23). Then, also $Q_k(s, a, o) = \mu \cdot Q_k(h(s, a, o))$ analogously to the proof of 2P-ZS-MG homomorphisms. □

3.21 Remark (New Aspects of Theorem 3.20)
One of the most interesting aspects of Theorem 3.20 besides its existence is that the proof for the case $\mu < 0$ needs the minimax theorem (Theorem 2.23) for matrix games. This

aspect should be highlighted because also an informal argumentation like "one exchanges the two agents in equal situations", which is often given by practitioners and points out the essence of the theorem, makes implicitly use of the minimax theorem. The reason is that the minimax theorem states that for an optimal pair of policies it does not matter whether agent one or agent two has to decide its strategy first (and telling it to the other) or whether both decide without knowledge of the other agent's policy. It is obvious that for such a property is important to speak about an "equal situation" of both agents.

3.22 Example (Robot Soccer, 4)
In Figure 3.1, some standard symmetries in robot soccer are depicted which should be respected by *any* model describing a robot soccer game. In this example, an abstract agent is a team consisting of two robots. A permutation of identical agents within a team is additionally possible but only depicted indirectly as the robots have no individual numbers.



(a) State s.

(b) State $g_y(s)$.

(c) State $g_x(s)$.

(d) State $(g_x \circ g_y)(s) = (g_y \circ g_x)(s)$.

Figure 3.1: Symmetries in robot soccer: a standard situation (state) s and its symmetric states $g_y(s)$ with exchange of the two teams, $g_x(s)$ without the exchange of teams, and the combination of both: $(g_x \circ g_y)(s) = (g_y \circ g_x)(s)$. The labels 1 and 2 indicate the team and the defended goal region, and the small black circle depicts the ball.

3.23 Remark (Fulfillment of the Projection Properties)
Any 2P-ZS-MG μ-homomorphism $h : \mathcal{SAO}_1 \to \mathcal{SAO}_2$ fulfills the projective properties (Equations 3.10, 3.13 and 3.14) because without using these properties it can be shown for $\mu > 0$ and $\mu < 0$ that
$\Pi_{s,a}(h^{-1}(h(s,a,o))) = \{(s',a') : \ f(s) = f(s') \text{ and } g_{s'}(a') = g_s(a)\}$,
$\Pi_{s,o}(h^{-1}(h(s,a,o))) = \{(s',o') : \ f(s) = f(s') \text{ and } i_{s'}(o') = i_s(o)\}$, and
$\Pi_s(h^{-1}(h(s,a,o))) = \{s' : \ f(s') = f(s)\}$.

Proof of e. g. $\Pi_{s,a}(h^{-1}(h(s,a,o)) = \{(s',a') : f(s) = f(s') \text{ and } g_{s'}(a') = g_s(a)\}$ for $\mu > 0$:
"$\subseteq$": Be $(s',a') \in \Pi_{s,a}(h^{-1}(h(s,a,o))$, i. e. there exists $o' \in \mathcal{O}(s')$ with $h(s',a',o') = h(s,a,o)$. Then $f(s) = f(s')$ and $g_{s'}(a') = g_s(a)$.
"$\supseteq$": Be $(s',a') \in \{(s'',a'') : f(s) = f(s'') \text{ and } g_{s''}(a'') = g_s(a)\}$. Then $f(s) = f(s')$ and $g_{s'}(a') = g_s(a)$. Choose an arbitrary $o \in \mathcal{O}_1(s)$ and $o' \in i_{s'}^{-1}(i_s(o)) \subseteq \mathcal{O}_1(s')$ which is non-empty since $i_{s'} : \mathcal{O}_1(s') \to \mathcal{O}_2(f(s')) = \mathcal{O}_2(f(s))$ is surjective. Then $h(s',a',o') = h(s,a,o)$ which means $(s',a') \in \Pi_{s,a}(h^{-1}(h(s,a,o)))$.
The other projections for $\mu > 0$ are similar. For $\mu < 0$, only the different image spaces, e. g. $i_{s'} : \mathcal{O}_1(s') \to \mathcal{A}_2(f(s')) = \mathcal{A}_2(f(s))$ have to be taken into account additionally.

3.24 Proposition (Composition of 2P-ZS-MG μ-Homomorphisms)
Let $h_1 : \mathcal{SAO}_1 \to \mathcal{SAO}_2$ be a μ_1-homomorphism and $h_2 : \mathcal{SAO}_2 \to \mathcal{SAO}_3$ be a μ_2-homomorphism for three associated 2P-ZS-MGs $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SAO}_1, T_1, R_1)$, $\mathcal{M}_2 = (\mathcal{D}_2, \mathcal{S}_2, \mathcal{SAO}_2, T_2, R_2)$, and $\mathcal{M}_3 = (\mathcal{D}_3, \mathcal{S}_3, \mathcal{SAO}_3, T_3, R_3)$. Then, $h_3 = h_2 \circ h_1 : \mathcal{SAO}_1 \to \mathcal{SAO}_3$ is a μ_3-homomorphism with $\mu_3 = \mu_1 \cdot \mu_2$.

Proof: To check that h_3 is a 2P-ZS-MG μ_3-homomorphism, one first must notice that the domains and co-domains of the component functions $f_k, g_{k,s}, i_{k,s}$ ($k \in \{1,2,3\}$) of all three homomorphisms fit together for all four cases (μ_1, μ_2 positive or negative) because a change of sign is equivalent to a change of the agents.
According to Remark 3.23 the equivalence classes $[(s,a,o)]_{h_3} \subseteq \mathcal{SAO}_1$ induced by h_3 fulfill the projection properties Equations 3.10, 3.13 and 3.14.
It must then be verified that the transition and reward functions also fulfill the 2P-ZS-MG μ_3-homomorphism conditions. For the transition function that means to proof that $\widetilde{T}_1(s,a,o,[s']_{h_3}) = T_3(h_3(s,a,o), f_3(s'))$ under the premise that $\widetilde{T}_k(s,a,o,[s']_{h_k}) = T_{k+1}(h_k(s,a,o), f_k(s'))$ for $k \in \{1,2\}$. The following equation is helpful in the proof

$$\begin{aligned}
[(s,a,o)]_{h_3} &= h_3^{-1}(h_3(s,a,o)) \\
&= (h_2 \circ h_1)^{-1}((h_2 \circ h_1)(s,a,o)) \\
&= h_1^{-1}(h_2^{-1}(h_2(h_1(s,a,o)))) \\
&= h_1^{-1}([h_1(s,a,o)]_{h_2}) \\
&= \bigcup_{(s',a',o') \in [h_1(s,a,o)]_{h_2}} h_1^{-1}(s',a',o')
\end{aligned}$$

because by projection holds $[s]_{h_3} = \bigcup_{s' \in [f_1(s)]_{h_2}} f_1^{-1}(s')$ which is needed below for $(*)$ together with the surjectivity of f_1:

$$\begin{aligned}
\widetilde{T}_1(s,a,o,[s']_{h_3}) &= \sum_{\widetilde{s''} \in [s']_{h_3}} T_1(s,a,o,s'') \\
&\overset{(*)}{=} \sum_{f_1(s'') \in [f_1(s')]_{h_2}} \sum_{s''' \in [s'']_{h_1}} T_1(s,a,o,s''') \\
&= \sum_{f_1(s'') \in [f_1(s')]_{h_2}} T_2(h_1(s,a,o), f_1(s'')) \\
&= \widetilde{T}_2(h_1(s,a,o), [f_1(s')]_{h_2}) \\
&= T_3(h_2(h_1(s,a,o)), f_2(f_1(s'))) \\
&= T_3(h_3(s,a,o), f_3(s')) \,.
\end{aligned}$$

Finally, the reward condition reads to $R_1 = \mu_1 \cdot (R_2 \circ h_1) = (\mu_1\mu_2) \cdot (R_3 \circ (h_2 \circ h_1))$ which shows that h_3 is a 2P-ZS-MG μ_3-homomorphism with $\mu_3 = \mu_1 \cdot \mu_2$. □

The intuition that the composition of 2P-ZS-MG μ-homomorphisms leads to coarser (if not equal) equivalence classes can formally be obtained by

$$[(s,a,o)]_{h_3} = \bigcup_{(s',a',o') \in [h_1(s,a,o)]_{h_2}} h_1^{-1}(s',a',o')$$

which is used in the proof of Proposition 3.24. Because h_1 is surjective for every $(s',a',o') \in [h_1(s,a,o)]_{h_2}$ there exists an $(s'',a'',o'') \in \mathcal{SAO}_1$ with $(s',a',o') = h_1(s'',a'',o'')$, i. e. $[(s,a,o)]_{h_3} = \bigcup_i [(s_i,a_i,o_i)]_{h_1}$ is a union of equivalence classes with respect to h_1.

3.25 Lemma (2P-ZS-MG μ-Automorphisms)
Let $\mathcal{M}_1 = (\mathcal{D}_1, \mathcal{S}_1, \mathcal{SAO}_1, T_1, R_1)$, be a 2P-ZS-MG with finite $\mathcal{SAO}_1$ and let $h : \mathcal{SAO}_1 \to \mathcal{SAO}_1$ be a 2P-ZS-MG μ-automorphism. Then $\mu \in \{1, -1\}$.

Proof: Let $r_{\max} = \max_{(s,a,o) \in \mathcal{SAO}_1} |R_1(s,a,o)| \neq 0$ be the maximal modulus of the reward of the 2P-ZS-MG. Since h is a 2P-ZS-MG μ-automorphism, $h \circ h = h^2$ is a 2P-ZS-MG μ^2-automorphism by means of Proposition 3.24 and $R_1 = \mu^2(R_1 \circ h^2)$. It must be verified that $\mu^2 = 1$. Two cases have to be excluded: $1 > \mu^2 > 0$ and $\mu^2 > 1$. If $\mu^2 > 1$, then all state-actions (s,a,o) with $R_1(s,a,o) = r_{\max}$ are mapped to a state-action pair with reward modulus $r_{\max} \cdot \mu^2 > r_{\max}$, in contradiction to the fact that h is an automorphism and $r_{\max}$ is maximal. If $\mu^2 < 1$, then each state-action (s,a,o) is mapped to a state-action with a modulus of reward $|R_1(s,a,o)| \cdot \mu^2 \leq r_{\max} \cdot \mu^2 < r_{\max}$, in contradiction to the fact that h is a bijection. □

3.26 Remark (Finite and Infinite 2P-ZS-MG μ-Automorphisms)
Lemma 3.25 shows that automorphisms of finite 2P-ZS-MGs can only have two forms: the first is the 2P-ZS-MG homomorphism ($\mu = 1$) and the second one can be considered as a 2P-ZS-MG homomorphism by exchanging the two agents ($\mu = -1$). For MDPs, the second possibility is meaningless and justifies the consideration of MDP homomorphisms without general $\mu > 0$ of [171].
However, for (countably and uncountably) infinite 2P-ZS-MGs the Lemma is *not* valid. Consider e. g. the following 2P-ZS-MG $\mathcal{M} = (\mathcal{D}, \mathcal{S}, \mathcal{SAO}, T, R)$ with $\mathcal{S} = \mathbb{Q}$ or $\mathcal{S} = \mathbb{R}$, trivial action sets $\mathcal{A}_s = \{a\}$, $\mathcal{O}_s = \{o\}$ for all $s \in \mathcal{S}_1$, a trivial transition function $T(s,a,o,s') = \partial_{s,s'}$ for all $s, s' \in \mathcal{S}$, and $R(s,a,o) = s$. Then, for every $\mu \in \mathcal{S} \setminus \{0\}$ a μ-automorphism is defined by $h(s,a,o) = (\mu \cdot s, a, o)$ because the complete structure (the trivial action spaces and transition function) is equal in every state and the reward is appropriately scaled.

Chapter 4

Supervised Learning (SL), Function Approximation, Generalisation

Contents

Wisdom is learning what to overlook.

William James (1842–1910)
(http://www.nonstopenglish.com)

Supervised learning (SL) is almost a synonym for function approximation. Sometimes the first one is considered to deal especially with noisy function evaluations and the second one with exact evaluations. There are two basic reasons for applying supervised learning (SL) techniques in RL: first, to simply compress an unmanageably large state space to a reasonable size assuming that the (unknown) underlying essential state space is much smaller than the standard one, and second to gain insight about a problem by discovering this essential state space. For both reasons the construction of *features*, i. e. a special kind of basis functions in the function approximation sense[(1)] which is often easy to interpret and inspired by human intuition, is one of the most common techniques besides neural network approaches. However, theoretically every set of basis functions such as radial basis functions of any kind, polynomials, or splines can serve as *function approximation architecture*.

This chapter is structured as follows: Section 4.1 gives a short introduction to SL and highlights the importance of automatically detecting good basis functions. In Section 4.2

[(1)] No orthogonality conditions are fulfilled and even the linear independency is often not checked in RL literature but fulfilled by trying to achieve the smallest set of features.

the key result on the provable convergence quality of value iteration in combination with SL methods is presented. Although the result improves a previous one in [19] it means that in practice it is very hard to guarantee convergence to an ε-approximation of the optimal value function since the approximation architecture has to provide very small errors in $\| \|_\infty$. Nevertheless, Section 4.3 provides an overview of practical results showing encouraging performance without any guarantee.

Numerical results of the author can be found in Section 5.2.6 in the context of other numerical studies.

4.1 Introduction

Supervised learning deals with finding a function $\widetilde{f} : X \to Y$ in some predefined function space $\mathcal{F}$ which approximates a given function $f : X \to Y$ in an optimal way with respect to some norm:

$$\widetilde{f} = \arg\min_{g \in \mathcal{F}} \|g - f\| . \tag{4.1}$$

Common norms are $\| \|_2$ or $\| \|_\infty$ which measure the mean and the maximum deviation, respectively. The more compact formulation

$$\widetilde{f} = \widetilde{\mathcal{A}} f \tag{4.2}$$

may be advantageous whenever $\mathcal{F}$ is undoubtedly defined. Generally, two types of errors occur: the *approximation error* due to the approximation architecture because $f \notin \mathcal{F}$, and the *estimation error* or *sample error* due to the finite number of sample data for which the function is evaluated [37, 157]. Sometimes, SL is understood to be function approximation with noisy data, i. e. the true function values $f(x)$ are modified by some additional noise, but throughout this thesis it is assumed to approximate noise-free function data.

Since the contraction property of the Bellman operators (contraction rate $\gamma < 1$) is valid for $\| \|_\infty$, this norm should also be used for function approximation ([75, 78], approximation scheme bases on [196]).[(2)] However, many SL methods find approximations in L_p-norms, hence, it is valuable to give approximation bounds for approximate value iteration with these norms being better than the $\sqrt[p]{|\mathcal{S}|}$ factor of the standard estimation for norm equivalence [151].

A huge number of function approximation methods exists which include but are not limited to (see [88]) neural network methods [19, 62, 122, 156, 176, 4, 82] [(3)], fuzzy logic [16, 105], cerebellar model articulation controller (CMAC) [2, 3, 83, 198, 201], classifier systems [57], and local memory-based methods [51, 145, 146]. By *racing* different approximation architectures can be tested and their parameters optimised [59, 131, 132]. The main idea is that most of the possible choices of parameters are discarded by a few tests and only the remaining ones are tested more intensively. A comparison of different approximation architectures in the sense of good fitting *and* a simple model (no overfitting) is studied in [121].

A subtle remaining question is *which* function(s) to approximate: one of the value functions (V or Q), or directly the policy π, see also Section 4.1.2. Approximating the state value

[(2)] An alternative would be to use weighted $\| \|_2$ norms [149] but the weights have to be determined first.

[(3)] Pesch states that some standard neural networks can lead to ill-conditioned optimisation problems [165] and gives further references.

function yields a smaller domain than the state-action value function, therefore it may be better to approximate with fewer parameters, but this comes at the cost of having to calculate the Q-values in each step. In 2P-ZS-MGs (not in MDPs) both approaches suffer from the fact that the value of a potentially large matrix game has to be determined for every evaluation of the policy. Approximating the policy is the only method which avoids matrix game calculations for an acting agent but is not suited for the process of determining the optimal policy because the evaluation of the value of a state(-action) would be very expensive (policy evaluation involves the *complete* state space). Even *policy search* methods [178] need to estimate the quality of several policies and therefore have to estimate value functions or rely on heuristics. Finally, it is also possible to approximate several or all three functions which can also be useful for mutual error tests.[(4)]

4.1.1 General Approximation Results

A qualitative result often mentioned as *curse of dimension* and *blessing of smoothness* is that with a degree of smoothness s and dimension of input parameter $\dim(x) = d$ the function $f(x)$ can only be expected to be approximated by an error of $O(n^{-s/d})$ where n is the number of parameters of the function approximation scheme [157]. [82] states that the curse of dimensions can not be broken by neural networks with multi-layer perceptrons, radial basis functions or similar nonlinear techniques. This means that neural network approaches can only be successful if a problem is simpler at the core.

4.1.2 Generalisation

Generalisation is one reason why supervised learning is employed. It means that the experience of a learner is transferred from the current situation to comparable ones. Generalisation is motivated by two facts: firstly, for large finite (or continuous) state spaces it is not possible to enumerate all states and, secondly, similar states often have similar values and optimal actions. Thus, generalisation is related to the two basic reasons for applying SL: state space reduction and a gain on insight.

Sometimes, generalisation and function approximation is separated artificially, however generalisation can always be interpreted in terms of function approximation. Some generalisation methods are used to determine a function $e_X : X \to \widetilde{X}$ such that the domain of $\widetilde{f}$ is no longer X but instead $e_X(X) \subseteq \widetilde{X}$, and are called *feature extraction* methods [19]. These can also be local [111] or action dependent [197] for Q-value functions. The hope is that if g is complicated enough that $\mathcal{F}$ can be chosen in a simpler way because the approximation is performed with $\widetilde{f} \in \mathcal{F} \circ e_X$.

Other types of generalisation methods do not make use of $f(x)$ but only of the distribution of inputs x_i of some input pairs $(x_i, f(x_i))$. These are collectively called *unsupervised learning* methods and can be interpreted as an input data preprocessing. Clustering approaches belong to this class. However, clustering approaches can also be applied to the $f(x_i)$ to extract all states with a similar value as a feature and hence become SL methods.

Generalisation in RL. As alluded to in the introduction, generalisation in RL can be divided into different approaches depending on the domain [88]. *Generalisation over states*

[(4)] [78], p. 413, shows how to perform value function updates computationally more efficiently if the transition function T is known by separately storing expectation values for each function of a linear approximation architecture.

such as feature extraction reflects the question of structural credit assignment: Which characteristica influence the optimal value function most? These methods basically reduce the description of the state space $\mathcal{S}$ by forming equivalence classes of similar states. Forming strict equivalence classes by exact model reduction is more restrictive but yields exact results (Chapter 3). Another strategy is to directly approximate the value function during value iteration [23, 206] or in Q-learning [108, 212]. In general, function approximation can affect the convergence of the iterative algorithms [23, 207] if one can not show that the approximation scheme is less expansive (in max norm) than the Bellman operator is contractive [72, 97]. Residual algorithms are one approach to address this problem [7] while some standard approaches like neural nets and linear regression are not theoretically justified even if working well in practice [72]. Instead, non-expansive approximation can be performed by *averagers* but these lack adaptivity [72]. Furthermore, adaptive resolution models, e. g. variable resolution dynamic programming (kd-tree [142]), the PartiGame algorithm [143], and decision trees [31, 134] can be seen as generalisation approaches.

Alternatively to states, state-action pairs could be subject to generalisation. For approximating Q-values, two approaches exist: the first is to use one approximation architecture for each action (if there are not too many different actions), the second is to employ only one architecture for all state-action pairs. The last method is the only possible approach for continuous action spaces. Training of continuous actions can be done by local gradient ascent methods [8], by modifying variance and mean of normal distributions of the actions to execute [80], or by application of *Kohonen maps* (*self-organising maps* [93]) which adaptively choose a discrete set of actions during learning [193][(5)].

4.1.3 Function Approximation with Automated Basis Determination

Choosing the approximating function space $\mathcal{F}$, i. e. *basis functions* in the case of linear function approximation, is an important issue.[(6)] There are three qualitatively different methods to choose the functions: first, one can directly choose finitely many functions $f_1, \dots f_n$, second, one can choose parametrised families of functions $f_{1,\alpha_1}, \dots f_{n,\alpha_n}$ and choose finitely many functions (perhaps with some optimisation over the parameters) which are most suited to approximate the given function, and third, one can try to construct the functions in a completely unparametrised way.

Obviously, the challenges increase from the first to the third method. Some work in the spirit of the last method is by Mahadevan [126, 128]). In his approach, so-called *proto value functions* serve as basis functions and are constructed only by state space topology information.[(7)] The second of the three methods is similar to feature detection and contains essential degrees of choice (parameters) for the approximation algorithm. Hence, this is also considered to be automated basis determination. Besides the predefined functions and free parameters the algorithms of the second type need a selection criterion which is in the best case an optimisation criterion. The two aims of this selection are the general ones of function approximation: to fit the function well and to avoid overfitting. *Cross validation* errors e. g. leave-one-out tests or more generally, the distinction of a training set and an evaluation set of points is an important method to reduce overfitting [146]. The

[(5)] The only problem is that the construction of appropriate Kohonen maps seems to be mathematically ill-posed.

[(6)] Linear function approximation means in this context the approximation by a linear sum of specified non-linear functions and not the approximation by piecewise linear functions.

[(7)] The question arises whether a simple grid soccer topology possesses enough structure to take advantage of it.

approach of [59, 131, 132] is algorithmically interesting to adapt the evaluation effort to the possible success of single parameter values: The evaluation criterion is computationally more intensive for promising candidates and less for weaker candidates.

4.2 Value Iteration with SL: Convergence Result

[23] provides a phenomenological classification of types of convergence which sounds amusing from a theoretical point of view but reflects the possibilities of "solutions" practitioners have obtained: good convergence (all iterates V_k are represented well, convergence to the exact optimal value function), lucky convergence (not all iterates V_k are represented well, convergence to the exact optimal value function), bad convergence (convergence to a non-optimal value function), and divergence. From a mathematical point of view only the "good convergence" is acceptable and hence a criterion is developed in the following under which this good convergence will occur.

For the sake of completeness the definition of classical value iteration (Definition 2.35) is repeated:

4.1 Definition (Value Iteration (2P-ZS-MG))
The following algorithm is called value iteration*: select $\varepsilon > 0$, choose an arbitrary initial guess $V_0 \in \mathbb{R}^{|\mathcal{S}|}$ for the (state-) value function, and determine iteratively $V_k = \mathcal{B}_{\mathrm{MG}} V_{k-1}$ for $k = 1, 2, \ldots$ until $\|V_{k+1} - V_k\|_\infty \leq \frac{\varepsilon}{2} \cdot \frac{1-\gamma}{\gamma}$.*

This algorithm *would* converge to V^*, and provide an $\frac{\varepsilon}{2}$-approximation for the value function estimate and an ε-optimal stationary policy (as for MDPs), *if* one assumes that $\mathcal{B}_{\mathrm{MG}}$ can be exactly calculated. In Section 2.4.2 the following results are anticipated and applied to the scheme of numerical value iteration, i. e. roughly speaking a numerical error of solving DPs is interpreted as an SL technique. This is possible because for the analysis of the algorithm it does not matter whether the approximation error is introduced by an SL method or by a different numerical technique.

For an error analysis similar to [19] (compare Remark 4.4), which does not seem to be widely available for 2P-ZS-MGs in the literature[8], the combination of the SL method with the operator $\mathcal{B}_{\mathrm{MG}}$ is interpreted as a numerical version of the operator and denoted by $\widetilde{\mathcal{B}}_{\mathrm{MG}}$. For an operator $\widetilde{\mathcal{A}}$ describing the application of an approximation architecture, $\widetilde{\mathcal{B}}_{\mathrm{MG}} = \widetilde{\mathcal{A}}\mathcal{B}_{\mathrm{MG}}$ because a value function – here stemming from the previously applied SL technique – is plugged into the Bellman operator and the result is again projected by the approximation architecture. In practice, the operator $\widetilde{\mathcal{A}}$ can not get the complete value function $\mathcal{B}_{\mathrm{MG}} V_k$ as input meaning that the approximation should be even worse than Lemma 4.2 suggests. However, for the readability of the theoretical result these technical subtleties are omitted.

The assumption of a (maximal) error of ε_1, i. e. $\|\widetilde{\mathcal{B}}_{\mathrm{MG}} V - \mathcal{B}_{\mathrm{MG}} V\|_\infty \leq \varepsilon_1 \|V\|_\infty$ for all $V \in \mathbb{R}^{|\mathcal{S}|}$, leads to the following lemma:

4.2 Lemma (Error of Numerical Value Iteration (2P-ZS-MG))
Let $\mathcal{B}_{\mathrm{MG}}$ be the Bellman operator and $\widetilde{\mathcal{B}}_{\mathrm{MG}}$ a numerical realisation (e. g. by combination with an SL technique) with $\|\widetilde{\mathcal{B}}_{\mathrm{MG}} V - \mathcal{B}_{\mathrm{MG}} V\|_\infty \leq \varepsilon_1 \|V\|_\infty$.[9] Let further be $\widetilde{V}_0 = V_0$,

[8] An exception which generalises Bertsekas' results to 2P-ZS-MGs is an extended draft version of [181].
[9] $\|\widetilde{\mathcal{B}}_{\mathrm{MG}} V - \mathcal{B}_{\mathrm{MG}} V\|_\infty \leq \varepsilon_1 \|V\|_\infty$ for all V means that the operator $\mathcal{B}_1 := \frac{1}{\varepsilon_1}(\widetilde{\mathcal{B}}_{\mathrm{MG}} - \mathcal{B}_{\mathrm{MG}})$ has operator

$V_k = (\mathcal{B}_{\mathrm{MG}})^k V_0$, *and* $\widetilde{V}_k = (\widetilde{\mathcal{B}}_{\mathrm{MG}})^k \widetilde{V}_0$ *the corresponding* k*-th value iterates. Then*

$$\|\widetilde{V}_k - V_k\|_\infty \leq \underbrace{\varepsilon_1 \cdot \sum_{i=0}^{k-1} \gamma^{k-i-1} \|\widetilde{V}_i\|_\infty}_{=\mathcal{E}_V(k)} \leq \frac{\varepsilon_1}{1-\gamma} \max_{i=0,\ldots,k-1} \|\widetilde{V}_i\|_\infty . \tag{4.3}$$

Proof: The case $k = 1$ simply reads to $\|\widetilde{V}_1 - V_1\|_\infty = \|\widetilde{\mathcal{B}}_{\mathrm{MG}}\widetilde{V}_0 - \mathcal{B}_{\mathrm{MG}}V_0\|_\infty \leq \varepsilon_1 \|\widetilde{V}_0\|_\infty$. It is now assumed, that the lemma is true for the value iterates of index k. Then

$$\begin{aligned}
\|\widetilde{V}_{k+1} - V_{k+1}\|_\infty &= \|\widetilde{\mathcal{B}}_{\mathrm{MG}}\widetilde{V}_k - \mathcal{B}_{\mathrm{MG}}V_k\|_\infty \\
&= \|\widetilde{\mathcal{B}}_{\mathrm{MG}}\widetilde{V}_k - \mathcal{B}_{\mathrm{MG}}\widetilde{V}_k\|_\infty + \|\mathcal{B}_{\mathrm{MG}}\widetilde{V}_k - \mathcal{B}_{\mathrm{MG}}V_k\|_\infty \\
&\overset{(*)}{\leq} \varepsilon_1 \cdot \|\widetilde{V}_k\|_\infty + \gamma \cdot \|\widetilde{V}_k - V_k\|_\infty \\
&\leq \varepsilon_1 \cdot \|\widetilde{V}_k\|_\infty + \gamma \cdot \varepsilon_1 \sum_{i=0}^{k-1} \gamma^{k-i-1} \|\widetilde{V}_i\|_\infty \\
&= \varepsilon_1 \sum_{i=0}^{(k+1)-1} \gamma^{(k+1)-i-1} \|\widetilde{V}_i\|_\infty .
\end{aligned}$$

$(*)$ is valid because $\mathcal{B}_{\mathrm{MG}}$ is a contraction with rate γ and the line below uses the induction hypothesis. The second estimate of the lemma is by means of the infinite geometrical series. □

Lemma 4.2 provides a bound independent from the value iterates, e. g. if the sequence $(\widetilde{V}_k)_k$ is monotonically decreasing and $\widetilde{V}_k \geq 0$. If $\widetilde{\mathcal{B}}_{\mathrm{MG}} = \mathcal{B}_{\mathrm{MG}}$, monotonicity is guaranteed as soon as $V_0 \geq \mathcal{B}_{\mathrm{MG}}V_0$ (analogously to the MDP case which e. g. is true if $V_0(s) = \frac{1}{1-\gamma} \max_{(s,a,o)\in\mathcal{SAO}} R(s,a,o)$ for all $s \in \mathcal{S}$ because then the Q-value iterate $Q_1(s,a,o) = R(s,a,o) + \frac{\gamma}{1-\gamma} \cdot \max_{(s,a,o)} R(s,a,o) \leq \frac{1}{1-\gamma} \cdot \max_{(s,a,o)} R(s,a,o)$ and Proposition 2.25, 2.) completes the argument).
A reference for the monotonicity result in 2P-ZS-MGs seems to be lacking and therefore the proof is sketched here: The first part is to show that $\mathcal{B}_{\mathrm{MG}}$ preserves $\geq$ (in the vector sense). This is seen by the assumption that we have a vector representing a value function estimate that is greater or equal than a second one in all components. This implies that the corresponding Q-value functions (Equation 2.49) have the same greater or equal property, and the result follows from the monotonicity property of matrix games (Proposition 2.25). The second part is completely analogous to MDPs [168]: Noticing that the monotonicity holds with $\mathcal{B}_{\mathrm{MG}}$ also for $\mathcal{B}^i_{\mathrm{MG}}$ for all $i \in \mathbb{N}$ leads directly to $V_{k+1} = \mathcal{B}^k_{\mathrm{MG}}(\mathcal{B}_{\mathrm{MG}}V_0) \leq \mathcal{B}^k_{\mathrm{MG}}(V_0) = V_k$.
However, for use in algorithms the tighter bound of the above lemma should be preferred because it can be computed iteratively (as suggested by the inductive proof: starting with $\varepsilon_1\|\widetilde{V}_0\|_\infty$ and assuming that the k-th step is already performed the sum has to be multiplied by γ and then $\varepsilon_1\|\widetilde{V}_k\|_\infty$ has to be added for obtaining the result in iteration $k+1$).

norm equal to or less than 1: $\|\mathcal{B}_1\| = \sup_{V\neq 0} \frac{\|\mathcal{B}_1 V\|_\infty}{\|V\|_\infty} \leq 1$. Furthermore, the above definition implies $\widetilde{\mathcal{B}}_{\mathrm{MG}} = \mathcal{B}_{\mathrm{MG}} + \varepsilon_1\mathcal{B}_1$.

4.3 Corollary (New Stopping Criterion for Numerical Value Iteration)
If the stopping criterion is changed to $\|\widetilde{V}_{k+1} - \widetilde{V}_k\| \leq c(\widetilde{V}_0, \ldots, \widetilde{V}_k)$ with

$$c(\widetilde{V}_0, \ldots, \widetilde{V}_k) = \Big(\frac{\varepsilon}{2} - \mathcal{E}_V(k+1)\Big) \cdot \frac{1-\gamma}{\gamma} \;-\; (\mathcal{E}_V(k+1) + \mathcal{E}_V(k)) \tag{4.4}$$

where the errors $\mathcal{E}_V(k)$ depend on the first $k-1$ numerical value iterates (Equation 4.3), and if the numerical approximation $\widetilde{\mathcal{B}}_{\mathrm{MG}}$ is a contraction of rate γ (like $\mathcal{B}_{\mathrm{MG}}$)[10] then the numerical approximation of value iteration also yields results comparable to the original value iteration.

Proof: Utilising notation of Lemma 4.2 the error of numerical and theoretical value iterates can be related by

$$\begin{aligned} \|V_{k+1} - V_k\|_\infty &\leq \|V_{k+1} - \widetilde{V}_{k+1}\|_\infty + \|\widetilde{V}_{k+1} - \widetilde{V}_k\|_\infty + \|\widetilde{V}_k - V_k\|_\infty \\ &\leq \|\widetilde{V}_{k+1} - \widetilde{V}_k\|_\infty + \mathcal{E}_V(k+1) + \mathcal{E}_V(k) \\ &\leq \Big(\frac{\varepsilon}{2} - \mathcal{E}_V(k+1)\Big) \cdot \frac{1-\gamma}{\gamma}\,. \end{aligned}$$

According to classical value iteration $\|V_{k+1} - V^*\|_\infty \leq \big(\frac{\varepsilon}{2} - \mathcal{E}_V(k+1)\big)$, hence

$$\|\widetilde{V}_{k+1} - V^*\|_\infty \leq \|\widetilde{V}_{k+1} - V_{k+1}\|_\infty + \|V_{k+1} - V^*\|_\infty \leq \frac{\varepsilon}{2}\,.$$

In prooving the quality of value function approximation it was not necessary to use that $\widetilde{\mathcal{B}}_{\mathrm{MG}}$ is a contraction of rate γ. However, for adopting the proof in [168] that the policy π_{k+1} induced by $\widetilde{V}_{k+1}$ is ε-optimal it is additionally needed that $\widetilde{\mathcal{B}}_{\mathrm{MG}}$ as well as the (linear) Bellman operator with fixed policy π_{k+1} are contractions with rate γ. □

Interpretation. The considerations above indicate that care should be taken (nothing can be guaranteed by Corollary 4.3) whenever the above defined numerical error ε_1 is in the same magnitude as the error ε of value iterates. If $\varepsilon_1 \ll \varepsilon$, γ not too close to 1, and $\mathcal{B}_1 := \frac{1}{\varepsilon_1}(\widetilde{\mathcal{B}}_{\mathrm{MG}} - \mathcal{B}_{\mathrm{MG}})$ fulfills a Lipschitz condition, then the solution of 2P-ZS-MGs can be performed nearly as well numerically as theoretically.

4.4 Remark (Comparison to Results of [19])
For the results in [19] about approximation quality of function approximation it is assumed that $\|\widetilde{\mathcal{B}}_{\mathrm{MG}}V - \mathcal{B}_{\mathrm{MG}}V\|_\infty \leq \varepsilon_1$, being independent from $\|V\|_\infty$. Analogously to Lemma 4.2, this leads to a (simpler) error bound of

$$\|\widetilde{V}_k - V_k\|_\infty \leq \varepsilon_1 \cdot \sum_{i=0}^{k-1} \gamma^i \leq \frac{\varepsilon_1}{1-\gamma}\,. \tag{4.5}$$

This error estimation could be utilised to define a different $\mathcal{E}_V(k)$ and to obtain a corresponding stopping criterion (Corollary 4.3).

4.3 Combination of RL and SL: Practical Results from Literature

Although Section 4.2 makes it clear that convergence of RL in combination with SL methods is hard to guarantee, some encouraging examples are to be presented in the following.

[10] If there exists some $n \in \mathbb{N}$ such that for all V, W holds $\|\mathcal{B}_1 V - \mathcal{B}_1 W\|_\infty \leq n\gamma\|V - W\|_\infty$ then $\widetilde{\mathcal{B}}_{\mathrm{MG}}$ is a contraction with rate $\gamma_1 \leq \gamma(1 + n\varepsilon_1)$, if $\gamma_1 < 1$, and therefore the weaker statement with discount factor γ_1 holds.

4.3.1 MDPs

In Section 4.1 the following incomplete list of function approximation methods is given and repeated here: neural network methods [19, 62, 122, 156, 176, 4, 82], fuzzy logic [16, 105], cerebellar model articulation controller (CMAC) [2, 3, 83, 198, 201], classifier systems [57], and local memory-based methods [51, 145, 146]. In addition to these references to special function approximation architectures, most of which already pointed to combinations with RL methods, the following reference is to be added: [177] combines neural networks in combination with some advanced Q-learning algorithms (e. g. $Q(\lambda)$ [164]).

Space continuous and Time Continuous Systems. Although it is in principle possible to apply function approximation methods to systems with continuous time (evolution by a differential equation) this approach suffers from the lack of a unique solution to the *Hamilton-Jacobi-Bellman equation* [152] if the concept of viscosity solution is not considered (analogous to differential games, Section 2.5). [35, 56] attack instances of space continuous *and* continuous time MDPs by means of the *Hamilton-Jacobi-Bellman equation* and function approximation. An exceptional example in which the concept of viscosity solutions is applied to RL algorithms for solving MDPs is [148]. A different approach in the time-continuous setting is a policy search, i. e. a search by gradients over a priori parametrised policies [150].

For a multi-agent system with many agents additionally to function approximation a decomposition of the value function into a sum of functions dependent on less agents (*factored representation*) can be helpful ([94, 76, 74, 162], or in combination with a suitable communication scheme: [79]). However, care has to be taken that even a factored reward and transition function does not imply a factored value function. The reason is that in each value iteration step the dependencies on other variables grow until at some point typically every variable is important for every state.

4.3.2 2P-ZS-MGs

Applications of function approximation in differential games include neural networks for driver assistance [96] and memory-based and kd-tree based methods for two-player pursuit evasion games [187]. Much of the work about the combination of discrete 2P-ZS-MGs and function approximation is done by Lagoudakis and concentrated on least squares policy iteration (LSPI): General results with an application to Littman's 1v1 grid soccer [97] and the utilisation of factored approaches for multiple agents [98] which are related to model reduction only with respect to actions[(11)], and an overview with a diversity of examples [99] are to be mentioned. [98] is quite close to the need of multi-agent robot soccer but due to the exponential dependence of the state space size on the number of agents the approach is not directly applicable because it only reduces the exponential dependence of the joint action space.

[(11)] The meaning of some formulae can be better understood by additionally considering [77].

Chapter 5

Robot Soccer and Other Applications

Contents

The complexity arises
when the principles are reduced to practice.

Robert F. Stengel
(Preface to [195])

The model of multi-player grid soccer is introduced and all important numerical results with respect to this model are presented in this chapter. The numerical results are broadly interpreted: different models and solution approaches such as MDP and 2P-ZS-MG models with or without symmetry reduction as well as DP and RL techniques and the impact of their parameters are numerically analysed. Furthermore, the resulting strategies are interpreted in a more soccer oriented fashion by goal rates per time and per team. Before giving a more detailed overview of the contents with references to single sections some

general comments on literature are to be made: A huge body of literature exists on the special application of robot soccer e. g. [69, 178, 197, 198] and references therein. For practical reasons, most of the work uses an MDP framework instead of an 2P-ZS-MG one which is at least questioned by the results presented in Section 5.2. The need for hierarchical structures for soccer with 11 versus 11 agents and the employment of heuristic ideas can often be observed [197, 215] as well as a simplification of robot soccer to keep away soccer [198, 215].

This chapter has the following structure: Section 5.1 summarises general thoughts of modeling the game of robot soccer and provides details of the special model "multi-player grid soccer" which serves as a basis for all following computations. The model is based on [112] but goes far beyond it because the dynamics for multiple agents have to be significantly changed. The numerical results are shown in Section 5.2 which possesses a rich substructure. Section 5.2.2 provides strong arguments for the choice of 2P-ZS-MG instead of MDP models in robot soccer, in Section 5.2.4 DP and RL techniques are compared with the result that exact and model exploiting DP techniques should be preferred whenever possible. These first evaluative numerical results appear natural but are surprisingly not standard in literature. DP methods are more intensively studied in Section 5.2.5: the dependency of convergence speed on different types of methods and parameters is numerically analysed. This includes comparisons of symmetry reduced models with their unreduced counterparts which is the practical application of the theoretical results in Chapter 3. A highlight is the "max-min convergence boosting phenomenon" which only seems to be present in 2P-ZS-MGs but not in MDPs and reveals unexpected "spontaneous" large error reductions during value iteration. Results with supervised learning (Section 5.2.6), a new DP algorithm which exploits almost invariant sets of the transition dynamics (Section 5.3), and general technical issues of applying strategies to real robots (Section 5.4), e. g. to the AIBO ERS-7, follow.

5.1 Modeling Robot Soccer

Robot soccer can be modeled for different purposes: the spectrum of possibilities ranges from a "true" continuous physical model including kinematic movements of every joint and object to a very coarse discrete model in which an elementary action already includes movement and ball-handling skills. A second distinguishing criterion is the mathematical range of models which is intertwined with the choice above: the model may be discrete or continuous (even with different physical degrees of modeling both options stay possible). Furthermore, the model may or may not include the policies of the opponent, and in the first case again with different degrees of freedom, e. g. the set of assumed opponent policies can be arbitrarily restricted.

5.1.1 General Issues of Modeling Robot Soccer

In this subsection some general thoughts which have an impact on every modeling of robot soccer shall be collected. It seems advantageous to apply the RL methods not directly with an initial value function $V_0 = 0$ to the robots and simply let the robots learn during play[(1)] some simplifications can be made even if the resulting model differs from reality.

[(1)] Even if the robot could detect its true state which is often difficult, the number of training steps can be quite large which leads to a vast amount of training time.

The obtained optimal policy of the approximate model can be understood as a very good initial guess of V_0 for an RL method applied to the real world problem.

This accounts for a general trade-off between model complexity and applicability of a model. On the one hand, the more precise the model is the better is the approximation for the "first guess" of the value function which can be improved stepwise by RL methods (a safe RL method in 2P-ZS-MGs is the WoLF (win or learn fast) method of [21]) to adapt to a special situation or opponent policy. On the other hand, the more unspecific a model is the more widely applicable it is, e. g. for different hardware realisation of the robots.

Continuous and Discrete Models, Stochasticity. Besides from technical difficulties of continuous game theoretic models (see Section 2.5 for an overview) these are typically considered to be deterministic. Clearly, introducing stochasticity to the continuous models would not simplify numerical computations. Thus, one can consider the choice to be only between a discrete stochastic and a continuous deterministic model.

There are several reasons why to prefer a discrete stochastic model for robot soccer. Firstly, (robot) soccer is a game with stochastic events not only because of unknown parameters such as the exact physical properties of the subsurface (grass or carpet) and unknown parameters in the robots' actuators and sensors but also because the movement of the ball kicked or hit by a robot is highly unpredictable. Secondly, the stochasticity of the model is also preferred because the decisions of the robots have to be stochastic. If the same situation always leads to the same deterministic reaction then the opponent team can exploit the observed behaviour in the next similar situation: e. g. if the position of a robot is slightly left to its opponent with respect to an appropriate axis this could lead to always going left around the opponent. Thirdly, the discretisation of the state and action space automatically leads to some abstraction away from a hardware specific model whereas a hardware specific model would be more appropriate for a differential game (the kinematics may be dependent on the robot type).

Symmetries. As mentioned in Chapter 3, respecting the inherent symmetry of a problem should be an issue. For example, in robot soccer the symmetries depicted in Figure 3.1 should be respected by any robot soccer model which assumes equal robots. One of the symmetries simply means to change left and right (from the point of view of the robots heading to a goal). This symmetry is only maintained if the robots possess the same symmetry and the abilities of the robots are symmetric. For human soccer, such a symmetry is clearly not assumable because it is standard to consider human players in the categories of preferring left, right or both. Only the last category of players fulfills the needs of that symmetry.

A second symmetry is due to the exchangeability of the two teams which is somewhat questionable. As remarked above the abilities of two teams can differ even if the robots do not. However, if one assumes that the two teams are about equally strong the assumption of exchangeability seems to be sensible *if* the model is not too detailed. This criterion is met e. g. by the model described in Section 5.1.2.

A third symmetry which is not depicted in Figure 3.1 is the permutation of robots within the same team. It relies on the equality of the robots and their abilities which is easily achievable for equal robots by using equal software. In terms of human soccer this means that every human player e. g. needs to have the strategical knowledge for every position (defense, center, attack) which is typically not valid.

Fair Kick-off Positions. A fair kick-off position after *every* goal is a distinguishing fea-

ture from standard human and standard robot soccer. Thus, it should be briefly discussed whether or not such a feature is to be employed in a model. First, in human and robot soccer the rules determine that the team against which a goal was scored gets the ball in the center of the soccer field (*kick-off position*). Only at the very beginning does one team get the ball at random and the other team gets it at the beginning of the second half. From a game theoric point of view even tossing coins and playing only one halftime is already fair (the probability distribution of start states $\xi \in \mathrm{PD}(\mathcal{S})$ has a value of $\sum_{s \in \mathcal{S}} \xi(s) V(s) = 0$). However, after tossing the coins the kick-off position typically is considered to be advantageously for the ball-possessing team which can also be verified for our later model (see Figure 5.3): e. g. for a 1v1 grid soccer the ball-possessing player state has a value of roughly 0.07 for a max-min optimal value function, i. e. even for a worst-case opponent there is a higher probability of winning for an optimally acting ball-possessing player.

Nevertheless, if a kick-off position is determined after *every* scored goal by chance this reduces the effect of a single random decision independently from the strength of each team. The standard rules improve the chance to win for a weaker team (against which probably more goals are scored), a fair kick-off position after every scored goal would increase the chance for the better team to win because they could randomly get the ball again even if they scored a goal directly before.

Additionally, the effect of a fair kick-off position after every goal can be interpreted as reducing the game to the expectation of scoring only *the first* goal which is sometimes used in soccer for avoiding a tie (golden goal rule). Time delaying strategies, e. g. keeping the ball and scoring a goal in the last minute to win and to avoid that the opponent scores a goal thereafter are neglected in such a model description. Instead, by means of the discounting factor early scoring is encouraged. For robot soccer, the repeated random criterion seems to be a sensible approach[(2)] even more because robots can – with the current battery configurations – play a full-time powerful game without any recreation phases which could be necessary for humans.

Concurrent Actions. Concurrent actions of the two teams seem very natural and essential in robot soccer. It is noted, however, because large parts of game playing literature are concerned with alternative turn games such as chess, checkers [179], or backgammon [205]. Some solution methods designed for the important class of deterministic alternative turn games, e. g. game tree search methods [118], are not feasible for stochastic games.

Abilities of the Robots. An important point is that in the following the hardware of all robots is considered to be equal. Of course, robots can have different abilities by different software but at least it is assumed that every robot *could* have the same skills. Consequences of these assumptions are that each robot is able to walk as fast as any other robot, has the same skills of ball handling, and the same communicational and computational performance features. For example, this condition is true for the AIBO league[(3)] in which AIBO ERS-7 robots without modifications have to be used, or for simulation leagues. However, in some leagues, especially if self-constructed robots are used, these ideal conditions are not given but instead a range of allowed performance features has to be assumed. Additionally, even for physically equal robots it may depend on the software of a team to which level the utilised skills deviate from best possible ones.

[(2)] Even for human soccer it could be interesting to encourage early goals to make games increasingly fascinating.

[(3)] The AIBO league is now called standard platform league because a new robot (NAO) has been introduced to the league.

Ball Possession. In general, the ball need not be close to any robot during a game. However, it can be considered sensible and typical that if no robot can control the ball directly, some of the next situated robots should try to get the ball. Hence, it is possible to model the ball so that it is always in the possession of one robot to reduce the number of states in the soccer game. Furthermore, in a sensible soccer model there have to be options for *dribbling* around an opponent and *passing* to a teammate.

Intra-Team Communication. It is theoretically important whether the team has a communication structure or not. The reason is that a team with perfect communication structure can be treated as a single (complex) agent making decisions while any imperfect communication structure must be treated as if each robot acts as a single agent with different information on the game. Fortunately, in robot soccer communication among a team is allowed such that the case of perfect communication can be assumed.

5.1.2 A Simple Multi-Player Robot Soccer Model

In this subsection the general thoughts of Section 5.1.1 are combined with practical issues to formulate a model of robot soccer which is abstract enough to be independent from hardware but specific enough to yield a meaningful policy for robot soccer. A main source of inspiration was [112]. Later on, this model is used for all numerical computations. Its key aspects are: the model is discrete and stochastic and respects some standard symmetries, all robots are equal, the ball is always in possession of one robot, and the kick-off positions are fair after scoring a goal – not only for the total halftime.

Furthermore, the standard order of different parts of actions during a time step is first dribbling, then moving, and finally passing. In real robot soccer, many basic abilities such as walking towards a predefined location, handling the ball (dribbling and kicking), and skills for the analysis of visual information (self and opponent localisation) are needed and highly non-trivial but assumed to be already available.

Now, the detailed model follows: The model is a 2P-ZS-MG $\mathcal{M} = (\mathcal{D}, \mathcal{S}, \mathcal{SAO}, T, R)$ where the decision epoch $\mathcal{D} = \mathbb{N}_0$ is scaled appropriately such that the time of performing one of the later described actions is approximately scaled to 1. In practice, actions can have different durations but the model is to be kept simple.

State Space. The state space $\mathcal{S} \subseteq \mathbb{N}^{2(n_a+n_o+1)}$ is discrete and consists of two dimensions for every robot in Team 1 and the opponent team the numbers of which are n_a and n_o, respectively, and an extra pair of dimensions for the ball.[(4)] Several robots can share the same position (i. e. grid box, grid cell) on the field because otherwise blocking an opponent would be too easy for a multiple robot game. The ball must share the position with some robot. More specifically, the grid resolution in the two dimensions will often be of the type (6×4), i. e. the position of each robot $(x_i, y_i) \in \{1, \ldots, 6\} \times \{1, \ldots, 4\}$, see Figure 5.1. An abbreviation will be "3v2-game" which means that $n_a = 3$ and $n_o = 2$. In terms of 2P-ZS-MGs each team is considered to be one of the two players of the Markov game which implicitly means that perfect communication is assumed.

Action Spaces. The action spaces $\mathcal{A}(s)$ and $\mathcal{O}(s)$ are constructed by the same principles to enable the "player exchanging symmetries". From the construction principles it becomes clear that all symmetries of Figure 3.1 are respected by $\mathcal{SAO}$. There are two different main

[(4)] The software package DRPOST maintains two different options for the ball: the coordinate version as shown here or the number of the ball-possessing robot which is more compact if the ball coordinates are limited to robot positions.

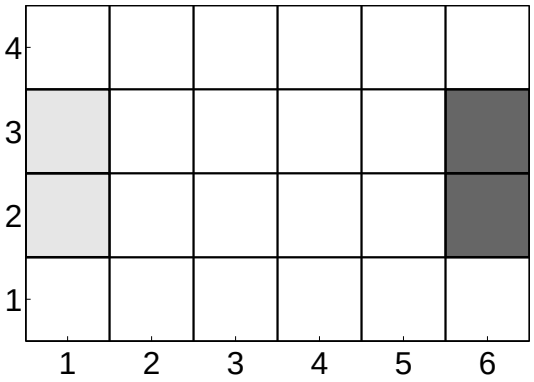


Figure 5.1: Discretisation of a soccer field: grid soccer. Dark grey indicates the defended goal region of the first team, light grey the defended goal region of the opponent team.

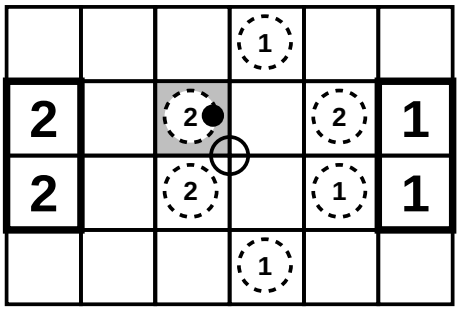
types of motion which can be alternatively performed in each time step: a move and a pass.

1.) *Moves*. In principle, for every robot of a team a move on the grid with *squared Euclidean distance* less or equal than 1 is allowed, i. e. it is possible to go one box N(orth), E(ast), W(est), S(outh) or to stand (0). Actions to leave the soccer field and diagonal moves are not allowed.[(5)] In contrast to real (robot) soccer, no robot can leave the soccer field and no fouls occur. Fouls could simply be integrated e. g. by assigning a foul probability which increases depending on how crowded a grid cell is.

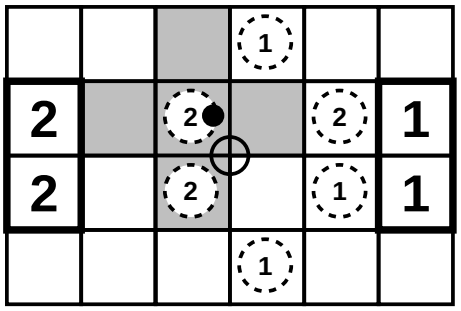
2.) *Passes*. Passes have only to be added to the action set if the team size is larger than 1 and if the ball-possessing robot is the only robot in its grid box. For a larger number of robots in the same box – even of the same team – it can not be guaranteed that the robots do not (unintendedly) obstruct each other. Furthermore, a pass is to be distinguished from a kick in the sense that it addresses a team member while a kick could also be an attempt to score a goal. However, kicks towards a goal are not modeled because it is assumed that they will automatically occur if a robot enters its opponent goal region.
Concerning a pass, there exist three parameters in the present model. The first parameter is called `max_kick_distance` which limits the maximum range of a kick (Figure 5.2). This parameter is the only one which influences the size of the action spaces and is standardly set to `max_kick_distance = 5` (squared Euclidean distance) for a (6×4)-grid. The other two parameters only affect the transition function and are described there. It is important to note that the position of the pass target is determined *after* its intended move.
It can be discussed whether a large kick range makes sense. An observation in robot soccer yields that for the AIBO league the robots are able to kick across the whole soccer field. The range is restricted in the model because in reality the reliability of a pass decreases with its distance.

Transition Function. The transition function has to be specified for every state-action pair. Since invalid actions such as leaving the soccer field or passing to team members being too far away are already filtered by the definition of the action spaces they need not be considered here. Furthermore, the description of the rules again reveals that they

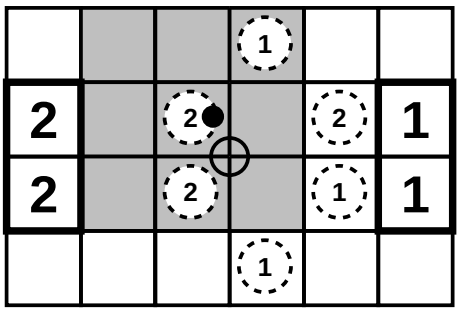
[(5)] The reason is not that the robots could not move diagonal but a diagonal move takes more time and is not reachable within one time unit. If desired, diagonal moves could be allowed which would dramatically increase the action spaces with the number of robots.



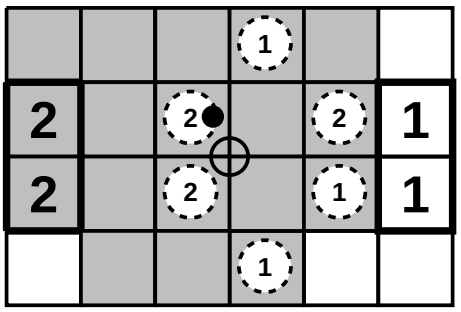
(a) `max_kick_distance` = 0.



(b) `max_kick_distance` = 1.



(c) `max_kick_distance` = 2.



(d) `max_kick_distance` = 5.

Figure 5.2: Grid soccer: the parameter `max_kick_distance` rules the maximum distance of the ball possessor to a team member being a pass target. Several values according to squared Euclidean distance are illustrated. `max_kick_distance` = 5 is the standard configuration of the present robot soccer model. As in Figure 3.1, the labels 1 and 2 indicate the team and the defended goal region, and the small black circle depicts the ball.

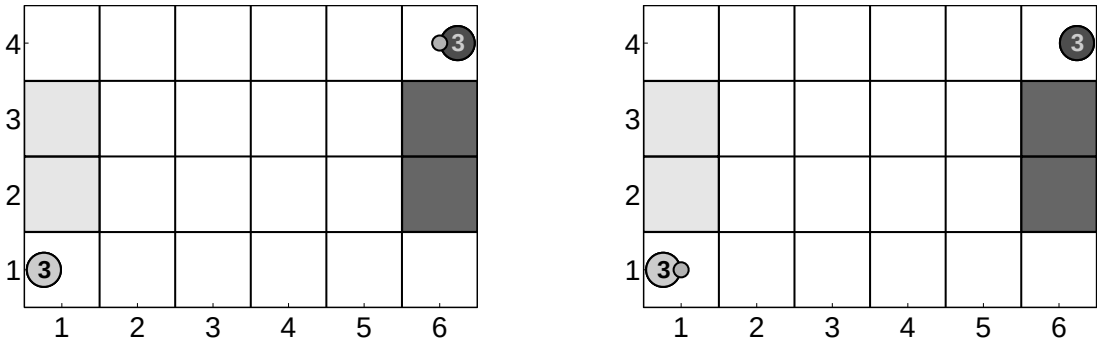
respect the symmetries of Figure 3.1 because there are no conditions which are different for the two teams or for a single robot. The standard order of different part of actions is dribbling, moving, and passing. Dribbling is only reflected in the ball possession and is not part of the action spaces. Dribbling and passing can not occur at the same time step: a passing robot is not allowed to move at the same time step, and the pass target (or a mistarget) gets the ball after moving.

1.) *Moves*. All movements of the robots are performed as intended but for the ball possession there are the following rules: If n is the number of all robots (of both teams) occupying the cell with the ball then the probability is $\frac{1}{n}$ for each robot to be on the ball after its movement. In soccer terminology this means that the dribbling phase is preceding the moving phase and that the dribbling phase is decided in every time step. It is imaginable that the previous action as well as the current action of each robot influences the probability of getting the ball but this would violate the Markov property. More importantly, the ball possession is not modeled actively: the robot which is on the ball in the last step has the same chance to keep it as every other robot in the same grid cell has to steal it. The model could easily be changed to give the ball possessor a different probability than a stealer but this would need a sound knowledge of the abilities of the robots.

2.) *Passes*. The soccer dynamics for the passes are more complicated than the movement dynamics because it seems to be essential that a pass can be intercepted by a robot not aimed at by the passing one. The reason is that a pass can be considered as a *risk option*: keeping the risk low and not passing moves the ball only a small amount towards the goal while a pass can move the ball a larger amount with a larger risk of a miskick (depending on the number of robots staying close to the pass target).
Concretely, three parameters exist in the model to influence the pass characteristics. The parameter `max_kick_distance` has been already discussed for the action sets and the remaining two, namely `close_distance` and `kick_good_prob_mult`, control the risk of a pass. `close_distance` determines how close an intercepting robot (team member or opponent) has to be such that it is a possible mistarget of the pass (the passing robot can also be a mistarget if close enough to the target, i. e. that it gets the ball again). It is assumed that the robot is not allowed to leave its grid cell towards the ball but it can get the ball.[(6)] `kick_good_prob_mult` determines how many times the probability is larger for performing the pass to the correct grid box. This means that *each* robot in the targeted box has the same (higher) probability – not only the targeted robot. The standard values for all computations (in 1v1 up to 3v3 soccer) for a (6×4)-grid are `close_distance` $= 2$ in squared Euclidean distance which is illustrated in Figure 5.4 and `kick_good_prob_mult` $= 3$. Again, note that the target position and closeness relations of the robots is determined by the positions *after* their intended move of that time step.
From a soccer perspective it seems to be sensible that dribbling and moving, only passing, or moving and recieving a pass are the three basic components of action. In robot soccer what may occur is that also teammates unintendedly also steal the ball because a crowded grid box can easily lead to a very uncontrolled outcome of dribbling actions. If controllability skills for the ball increase it will be possible to change the model to one in which the probability of keeping the ball is higher or in which one team randomly gets the ball and the robots can decide which teammate gets it.

[(6)] In practise, it is assumed that the robot starts some intercepting behaviour if the ball comes close enough to its location. However, the interception distance is much smaller than the distance of one grid cell because the ball moves too fast for a long interception procedure.

3.) *Kick-off*. At the very beginning as well as after every scored goal a fair kick-off position (see Section 5.1.1) is initialised. Particularly, the two states in Figure 5.3 are assigned a probability of $\frac{1}{2}$. Since they are agent exchanging symmetric (agent = team) they are fair according to Theorem 3.20. For value iteration it is only important that the stochastically weighted sum of values for these positions is zero. However, for RL methods it can make a big difference in the speed of learning if an equal weighting of the two positions of Figure 5.3 or an equal distribution of all states in $\mathcal{S}$ is chosen. For example, the latter can encourage exploring different regions of the state space without any exploration strategy in the RL method.



(a) Kick-off state s_1.

(b) Kick-off state s_2.

Figure 5.3: The two kick-off states of multi-player grid soccer, here 3 versus 3 players (3v3 grid soccer). Dark grey indicates the robots and defended goal region of the first team, light grey the robots and defended goal region of the opponent team. In contrast to other figures before, the numbers do not indicate the team but the numbers of robots of that team being in the same grid cell. The small grey circle represents the ball which is not attached to a special robot in that cell.

In general, due to the nature of the stochastic transition function it is easy to model a mis-action of any kind by giving a low probability to an action not intended by the player. In this way, (small) sensor errors can be modeled. Since it is assumed that all robots have the same degree of errors no team has an advantage over the other. Therefore, it is assumed that neglegting (small) errors has little impact on the optimal policies.

5.1 Example (Robot Soccer, 5)
A special situation (Figure 5.4) should illustrate the pass mechanism described above. One of the robots of Team 2 stops and plans a kick to a teammate indicated by a large arrow. The kick is possible because the robot is not disturbed by a second robot in the same cell and the distance criterion marked by the dark grey area (partly overlapped by the light grey area) is satisfied. All robots are depicted after their planned movement of the time period which is shown by the small arrows. The possible mistargets are all robots in the light grey area, in this case only two robots of Team 1. Assuming that `kick_good_prob_mult` = 3, this results in a probability of $\frac{3}{5}$ for the intended pass and a probability of $\frac{1}{5}$ for each of the two mistargets. If the upper of the two robots of Team 1 moved to the target field then that robot and the intended target would have probabilities of $\frac{3}{7}$ while the second mistarget has a probability of $\frac{1}{7}$ for receiving the pass.

Reward Function: Neutral, Defensive, Aggressive Policy. Scoring a goal. The reward for Team 1 for entering the opponents goal region is 1 *if* the ball is also in that field

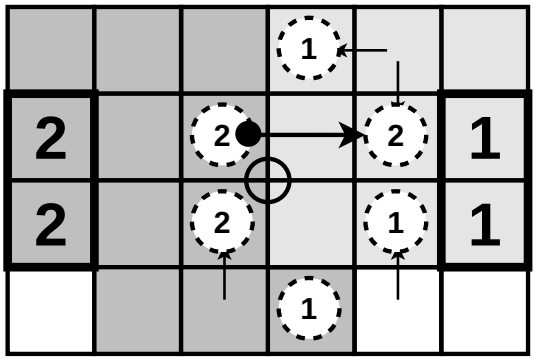


Figure 5.4: Grid soccer: the parameters `max_kick_distance` = 5 rules the maximum distance of the ball possessor to a team member being a pass target (dark grey boxes, partly hidden by light grey boxes), and by `close_distance` = 2 the possible mistargets are all other robots in the light grey boxes. The targeted grid box for the pass is indicated by the big arrow. Small arrows indicate the performed moves (no arrow means the action "stand") during the time period. As in Figure 3.1, the labels 1 and 2 indicate the team and the defended goal region, and the small black circle depicts the ball (see also Example 5.1).

(no matter which robot took the ball to that field) *and if* additionally at least as many offenders as defenders of the opponent team are in the same of the two cells of the goal region (Figure 5.1). It is assumed that otherwise the defenders successfully defend their goal (region) and a performed kick towards the goal is blocked. The corresponding agent exchange symmetric situation that Team 2 enters the goal region of Team 1 with the ball and a less or equal number of defenders of Team 1 is rewarded by -1 (scored goal against Team 1).

All other cases. The reward function simply needs to be 0 if no goal is scored because no team should gain an advantage by doing nothing. As discussed in Section 5.1.1 at "fair kick-off positions" the present score of a game is neglected in the model, which considerably reduces the number of states. This shortcoming could be relativised in real robot soccer by introducing three types of policies: neutral, defensive or aggressive which are employed if the score difference is equal, positive, or negative, respectively. The three different policies can be calculated (by sacrificing the team exchanging symmetry in the non-equal cases) with three different reward functions: scoring a goal is artificially ranked equal (as is described above), lower (smaller reward) or higher (larger reward) than letting the opponent score a goal. If not stated otherwise, equal ranking is assumed.

5.1.3 Symmetry

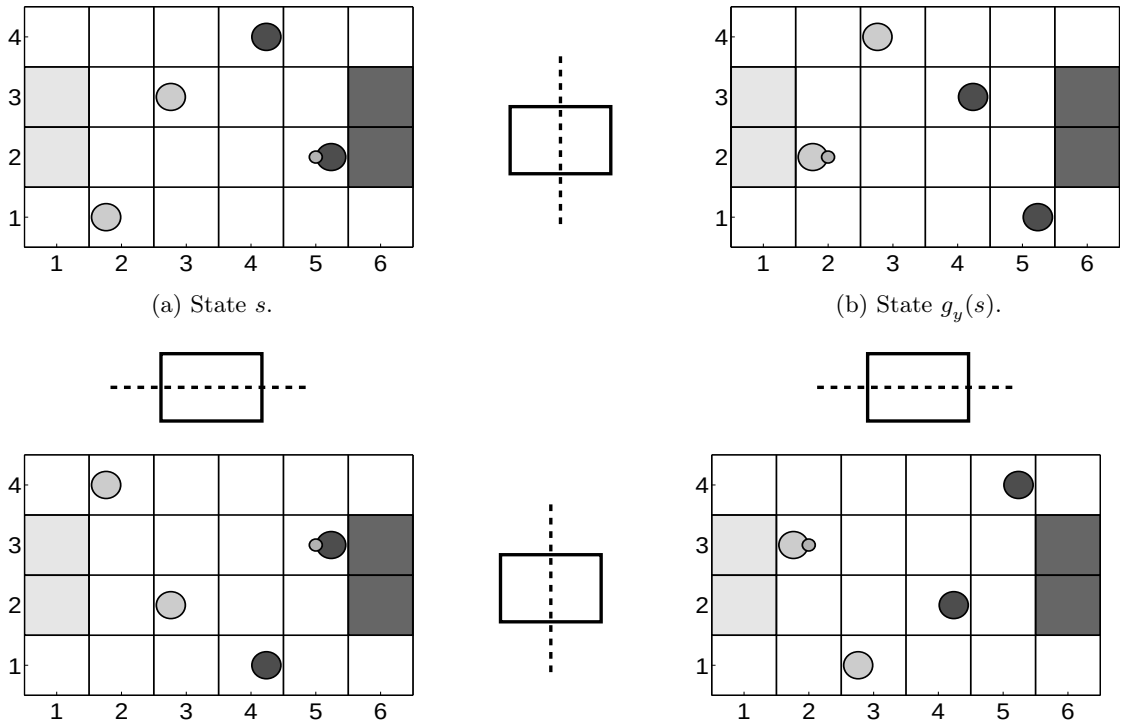
The multi-player grid soccer model of Section 5.1.2 is constructed in concordance with the general three symmetries mentioned in Example 3.22 which should be respected by any model of robot soccer:

1.) permutations of the players in the same team,

2.) reflection at the goal-to-goal line (g_x in Figure 3.1), and

3.) reflection at the mid-line and exchange of teams (g_y in Figure 3.1).

This is best illustrated by "a discretised version" of Figure 3.1 which is depicted in Figure 5.5.



(a) State s.

(b) State $g_y(s)$.

(c) State $g_x(s)$.

(d) State $(g_x \circ g_y)(s) = (g_y \circ g_x)(s)$.

Figure 5.5: Symmetries in grid soccer (discretisation of states in Figure 3.1): a standard situation (state) s and its symmetric states $g_y(s)$ with exchange of the two teams, $g_x(s)$ without the exchange of teams, and the combination of both: $(g_x \circ g_y)(s) = (g_y \circ g_x)(s)$. The small grey circle depicts the ball.

Utilising these symmetries for model reduction, the effect on the size of the state (and hence state action) space can be enormously (Table 5.1). In a multi-agent grid soccer, some symmetries help to reduce the state space[7] by a constant factor (reflexions), while others (robot permutations) are of increasing usefulness with a growing number of robots. Note that the third symmetry is particular to 2P-ZS-MGs and can only be applied for a soccer game with equally many robots in each team (an "xvx" game). This explains a qualitative difference of the reduction factors a/b in Table 5.1 which are roughly $(2 \cdot 2 \cdot n_a! \cdot n_o!)$ in special cases and otherwise $(2 \cdot n_a! \cdot n_o!)$. Applying these formulae for a grid size of (6×4), theoretically in a 6v6 grid soccer game the savings would be of the order of 10^6 and in the 11v11 version of about 10^{15} (yet, of course, even the reduced state space is still way too large).

Game	Standard (a)	Reduced (b)	a/b	Sym. 1	Sym. 2	Sym. 3
1v1	1152	282	4.1	1.0	2	2
1v2, 2v1	41472	10244	4.0	2.0	2	1
2v2	1327104	82944	16.0	4.0	2	2
3v2, 2v3	39813120	1742400	22.8	11.4	2	1
3v3	1146617856	8820000	130.0	32.5	2	2

Table 5.1: Number of states for different multi-player soccer games on a (6×4) grid in a standard and a symmetry reduced form. n_ov$\,n_a$ means a game with team size n_a for Team 1 and size n_o for team 2. While a/b denotes the total reduction factor by all symmetries, the last three columns clarify the effect of each single symmetry, the number of which corresponds to the enumeration list at the beginning of Section 5.1.3).

5.2 Numerical Results of Grid Soccer

The numerical results form an important part of the present work. They are organised as comparative studies including but also going far beyond the application of the theoretical results of Section 3.2. In the small amount of literature on 2P-ZS-MGs only comparisons of different max-min RL methods seem to be available but the key model design question of whether to consider max-min or max methods or whether DP or RL methods give higher performance are typically unanswered.

As a first answer it is clear that DP methods should be more effective than RL methods because the knowledge of the model is only available to the first class of algorithms. Therefore, the argument for RL methods is often not effectiveness but adaptiveness to an unknown model (partly unknown or slowly changing environments). However, it can be observed by the studies in Sections 5.2.2 and 5.2.4 – and is also known to the RL community – that applying the theoretical convergence results of RL methods to practical problems leads to uncomfortably large numbers of learning steps which is often unmanagable to perform in practice.

Many methods have been designed to speed up learning: some of them targeting at the update rules of the algorithms (such as multi-step updating or multi-state updates (function approximation)), others aiming at external knowledge directly (such as imitating humans or experts made by humans) or indirectly (such as hierarchical models with a hierarchy

[7] The size of the state space of a n_av$\,n_o$ game with a total of $n = n_a + n_o$ robots is $|\mathcal{S}| = n(6 \cdot 4)^n$ if the ball is uniquely assigned to one of the n robots and the grid size is (6×4).

specified by humans). However, the ultimate application of learning theory to design any method which is suitable for most models and is learning completely automatically has not yet been reached to the knowledge of the author.

The author tries to make a small contribution by incorporating DP methods which are neglegted by many RL researchers or only used for computation of the exact values of policies to compare final results of small models. One mission of this work is to claim that using DP methods with a non-perfect but simple model to construct a *first guess* approximation for the value function and then starting RL methods with this initial guess is an appropriate methodology. It is also imaginable that the fact that RL methods do not need to cover the whole state space $\mathcal{S}$ can be incorporated by simulating an RL trajectory and then constructing an initial guess by DP methods only on the states of this trajectory and on suitable additional states.

5.2.1 Preliminaries for the Following Subsections

The numerical results presented in Section 5.2 are computed by a software package called DRPOST (Discrete Robust Probabilistic Optimal Strategy Tool) which is implemented by the author as a bundle of MATLAB routines. These routines contain a model generating collection (for state and action spaces, transition and reward function) for a general n-player grid soccer model as described in Section 5.1, all considered DP and RL methods for use with MDPs and 2P-ZS-MGs, and a large amount of other functions which are needed, e. g. for function approximation, symmetry reduction, and for solving matrix games (see Appendix C).

Most of the following comparative studies are performed with a 1v1 soccer model on a 6×4 grid field if they are not explicitly intended to compare a single agent with a multi-agent team grid soccer model or to compare different sizes of the field. The reason is that the computation of the optimal value functions and strategies for a 2v2 grid soccer model on a small grid or a 1v1 grid soccer model on a larger grid needs too much time to compare large varieties of parameters. Additionally, many basic effects can be observed for the small model.

Standard parameters for the algorithms are a discount factor $\gamma = 0.9$ and an accuracy of $0.5 \cdot 10^{-3}$ for value functions in value iteration implying an accuracy of $\varepsilon = 1 \cdot 10^{-3}$ for the policy (Corollary 4.3). For RL methods a decayed learning rate $\alpha_n(s,a) = \frac{1}{n}$ (for each $s \in \mathcal{S}$) which meets the standard convergence criteria is used and the random exploration rate of the employed ε_1-greedy policy is $\varepsilon_1 = 0.2$. Since Q-Learning is an off-policy method the learned strategy is greedy independently from the policy followed during the learning phase. Furthermore, the standard number of learning steps is 100 (times $|\mathcal{S}|$) which is higher than any number of steps needed for convergence in DP methods. This number was chosen on the basis of the results of Figure 5.6.

Nomenclature. In the following, "team 1", "player P_1", or simply "the player (team)" is considered to be the robot soccer team which is controllable, e. g. by optimal policies computed by DP or RL methods. "team 2", "player P_2", or "the opponent (team)" is the collection of agents which are trying to score against team 1. In some of the comparisons, one player is considered to be omniscient, i. e. the strongest possible or worst-case opponent that already knows the policy of team 1, in some cases it is assumed that each team does not know the policy of the other team. To make the DP operators part of the description the DP and RL methods for classical MDPs are called max methods and for 2P-ZS-MGs

max-min methods according to the Bellman operator.[8]

There is also a nomenclature introduced to describe the policies efficiently without the need for giving action probability distributions for every state which would be hard to interpret. Table 5.2 shows relevant abbreviations (where the max method yields an element of the game theoretic best response to the other team's strategy) and the description of the table makes the reader familiar with the notation of e. g. $\mathrm{MM}_{\mathrm{VI}}(\mathrm{R})$.

Abbreviation	Meaning
M	strategy determined by max method
MM	strategy determined by max-min method
QL	strategy determined by Q-learning method
R	random strategy
VI	strategy determined by value iteration method

Table 5.2: Different abbreviations for special policies. Example: $\mathrm{MM}_{\mathrm{VI}}(\mathrm{R})$ means a policy that is determined by a max-min value iteration method against a random opponent. This is the only example in which the policy of the opponent does *not* influence the algorithm for determining the policy. The opponent's policy is crucial especially in all max methods and also in all RL methods.

Evaluation of Policies. In general, the quality of a policy π_1 for Team 1 is quantified in a simple and precise way by its value $V(\xi, \pi_1, \pi_2)$ which depends on an initial probability distribution ξ of start states[9], and the two policies π_i of the teams $i = 1, 2$. In the following $V(\xi, \pi_1, \pi_2)$ is often abbreviated by $V(\pi_1, \pi_2)$ and it is then assumed that ξ is the standard distribution of start states. If π_2 is also omitted it is considered to be a best response to π_1, i. e. $\pi_2 \in BR_2(\pi_1)$. Typically, the value of policies is determined to an accuracy of $\varepsilon = 1 \cdot 10^{-3}$.

Since the value of a policy may be hard to interpret in robot soccer, additional characteristics of the policies extracted by long-term simulations (many kick-off positions) are given: the fraction of total scored goals of both teams divided by the number of time steps as well as the percentage of goals of team 1. The first characteristic is intended to show how offensive or defensive the combination of policies is and the second one shows how successful each team is in comparison to the other. For example, if both characteristics are high then Team 1 probably has a successful offensive strategy while Team 2 has an unsuccessful defensive or offensive strategy. If the number of total goals is low but the success rate for Team 1 is high then Team 1 has a successful defensive strategy and Team 2 an unsuccessful defensive or offensive strategy. One problem remains: the characteristics may yield no results about an unsuccessful team strategy or if both teams are equally good. Therefore, in Tables 5.9 and D.12 each policy is evaluated in a simulation against a random opponent and against itself to obtain a measure of how offensive or defensive each policy is (the total number of scored goals may indicate this). In general, however, some care has to be taken about the precision of the simulation: although $1 \cdot 10^6$ simulation steps are performed for every policy the more total goals are scored during the simulation the more accurate are the goal statistics. As a conservative rule of thumb, the goal rate of player P_1 should not be argued about for absolute differences of ± 0.05.

[8] The software DRPOST also provides the option to choose the type of Team 1 by similar declarations.

[9] ξ is fixed in robot soccer to the distribution which assigns 50% probability to each of the two kick-off states of Figure 5.3.

5.2.2 Reasoning for 2P-ZS-MG Modelling: Comparison of MDP and 2P-ZS-MG strategies

The first subsections of the numerical results serve to reason for the authors choice of model and methods. The two basic statements are: for modelling it is preferable to use 2P-ZS-MGs instead of MDPs and for computing optimal strategies DP methods are important to initialise RL methods whenever this is possible. The most obvious theoretical argument for the first statement is that for an MDP a *deterministic* optimal policy always exist. This is typically determined by DP and RL algorithms and it is a best answer only to a fixed opponent policy. In contrast, the 2P-ZS-MG model a priori assumes a worst-case i. e. tactically strongest opponent which already knows the policy of the player. This has two advantages: because the computed policy of the player is safe against *any* strategy of the opponent it is first sufficient to compute *only one optimal policy* and not one against each of the infinitely many possible opponent policies and, second, it is *not necessary to know the opponent policy.*[(10)] In the opinion of the author the second advantage is more important because for relativising the first one it can be argued that a strong policy could be strong against a whole set of opponent policies.

A First Comparison

Tables 5.3 and 5.4 provide the result that a best response policy (π_1 by an RL and π_2 by a DP method) against a randomly acting opponent is easily outperformed by its own best response answer (π_3, π_4, and π_5).[(11)] This is not surprising because of the mainly deterministic nature of the max-optimal policy – only exact ties of the Q-values lead to randomised actions. Another observation is that with a higher number of learning steps RL methods in principle do as well as DP methods (π_3 is equally strong against π_1 as π_4 because the value is equal) indicating that the inherent problem lies in the non-stochasticity of the policy. This can be seen at first glance in Table 5.3 by noticing that also the theoretically best policy π_2 against a random opponent is easily outperformed by the worst-case opponent strategy π_5 which is reflected in the high value $V(\pi_5, \pi_2)$. The fact that the value $V(\pi_5, \pi_2)$ is equal to that of $V(\pi_4, \pi_1)$ shows that "bad learning"[(12)] of the RL method is not the reason for bad performance against a worst-case opponent.

To conclude the first comparison the analogue tables for max-min methods are presented which show that the max-min optimal policy can not be exploited – as expected – even by a worst-case opponent knowing this policy in advance.

By comparison of the small size 6×4 soccer field with the medium size 12×8 one it becomes obvious that RL methods can degrade with the number of states if the number of training steps are kept linearly related to the state space size $|\mathcal{S}|$. Max-min learning methods are specifically concerned: with the 100 (times size of the state space) learning steps the max RL methods approximate the optimal policy quite well (all values are equal in Table 5.4), however, for the max-min strategy the performance against a worst-case opponent degrades significantly. The difference between $V(\pi_4, \pi_1)$ and $V(\pi_5, \pi_2)$ of 0.074 in value seems not too dramatic at a first glance but the fact that the goal rate of the

[(10)] An RL method can adapt to a stationary (temporarily fixed) policy of other agents but problems occur if all agents are learning, i. e. changing their policies over time.

[(11)] Best responses are max optimal policies and in the case of RL methods ($\mathrm{M}_{\mathrm{QL}}(\cdot)$) only a more or less well approximation to a true best response $\mathrm{M}_{\mathrm{VI}}(\cdot)$.

[(12)] The RL policy is typically initialised as R(andom) such that for states never reached by the finite simulation the random action selection is kept.

	$\pi_3 = \mathrm{M}_{\mathrm{QL}}(\pi_1)$	$\pi_4 = \mathrm{M}_{\mathrm{VI}}(\pi_1)$	$\pi_5 = \mathrm{M}_{\mathrm{VI}}(\pi_2)$
$\pi_1 = \mathrm{M}_{\mathrm{QL}}(\mathrm{R})$	$V:$ 0.415 $g_t:$ 0.080 $g_1:$ 0.735	$V:$ 0.415 $g_t:$ 0.080 $g_1:$ 0.731	
$\pi_2 = \mathrm{M}_{\mathrm{VI}}(\mathrm{R})$			$V:$ 0.415 $g_t:$ 0.080 $g_1:$ 0.730

Table 5.3: Robustness of max-policies against worst-case opponents (6×4 soccer field). The column policies are that of player P_1 and the row strategies of player P_2. The value V and the relative amount of goals g_1 are from the view of P_1 (as always), whereas the total goal rate per time step g_t relates the sum of goals of both players to the number of simulated time steps.

	$\pi_3 = \mathrm{M}_{\mathrm{QL}}(\pi_1)$	$\pi_4 = \mathrm{M}_{\mathrm{VI}}(\pi_1)$	$\pi_5 = \mathrm{M}_{\mathrm{VI}}(\pi_2)$
$\pi_1 = \mathrm{M}_{\mathrm{QL}}(\mathrm{R})$	$V:$ 0.145 $g_t:$ 0.030 $g_1:$ 0.719	$V:$ 0.145 $g_t:$ 0.030 $g_1:$ 0.723	
$\pi_2 = \mathrm{M}_{\mathrm{VI}}(\mathrm{R})$			$V:$ 0.145 $g_t:$ 0.029 $g_1:$ 0.725

Table 5.4: Robustness of max-policies against worst-case opponents (12×8 soccer field). The explanation of how to read the table is as in Table 5.3.

opponent increases from the best of 50% to over 90% should give the correct impression (Table 5.6).

	$\pi_3 = \mathrm{M}_{\mathrm{QL}}(\pi_1)$	$\pi_4 = \mathrm{M}_{\mathrm{VI}}(\pi_1)$	$\pi_5 = \mathrm{M}_{\mathrm{VI}}(\pi_2)$
$\pi_1 = \mathrm{MM}_{\mathrm{QL}}(\mathrm{R})$	$V: 0.010$ $g_t: 0.070$ $g_1: 0.509$	$V: 0.010$ $g_t: 0.071$ $g_1: 0.505$	
$\pi_2 = \mathrm{MM}_{\mathrm{VI}}(\mathrm{R})$			$V: 0.000$ $g_t: 0.070$ $g_1: 0.502$

Table 5.5: Robustness of max-min-policies against worst-case opponents (6×4 soccer field). The explanation of how to read the table is as in Table 5.3.

	$\pi_3 = \mathrm{M}_{\mathrm{QL}}(\pi_1)$	$\pi_4 = \mathrm{M}_{\mathrm{VI}}(\pi_1)$	$\pi_5 = \mathrm{M}_{\mathrm{VI}}(\pi_2)$
$\pi_1 = \mathrm{MM}_{\mathrm{QL}}(\mathrm{R})$	$V: 0.074$ $g_t: 0.007$ $g_1: 0.979$	$V: 0.076$ $g_t: 0.008$ $g_1: 0.921$	
$\pi_2 = \mathrm{MM}_{\mathrm{VI}}(\mathrm{R})$			$V: 0.000$ $g_t: 0.023$ $g_1: 0.499$

Table 5.6: Robustness of max-min-policies against worst-case opponents (12×8 soccer field). The explanation of how to read the table is as in Table 5.3.

Training Against Better Opponents

To provide additional weight for our hypothesis above that the non-stochasticity of optimal policies for MDPs is the reason for the failure of these policies against a learning opponent, better training partners are chosen and the effect is studied. In the first comparison the policies π_1, π_2 of Table 5.3 are trained against a randomly acting opponent which can be considered to be very weak – only an opponent helping the player could be weaker. Therefore, in the following comparison in Tables 5.7 and D.10 the policies π_1, π_2 are trained against much better initial opponent policies which in fact are π_1 and π_2 of Table 5.3. The result is – by comparing Tables 5.3 and 5.7 – that the better initial training policy *increases* the exploitability by a best response strategy. Although this can be expected by means of the lack of stochasticity of the better training partner it questions efforts to present a strong policy to an MDP learner and ignoring the 2P-ZS-MG nature of robot soccer.

Exploitability of Non-Optimal Opponents

The most reasonable counter argument against using max-min methods is that they do not fully exploit weaknesses of the opponent's policy. This is correct for non-optimal and particularly for very weak opponents as can be seen by the case of exploiting a randomly acting opponent on a 6×4 soccer field in 1v1 soccer: the values of start states for the max policy is much higher than that of the max-min policy ($V(\xi_{\mathrm{start}}, \mathrm{M}_{\mathrm{VI}}(\mathrm{R}), \mathrm{R}) \approx 0.688 >$

	$\pi_3 = \mathrm{M}_{\mathrm{QL}}(\pi_1)$	$\pi_4 = \mathrm{M}_{\mathrm{VI}}(\pi_1)$	$\pi_5 = \mathrm{M}_{\mathrm{QL}}(\pi_2)$	$\pi_6 = \mathrm{M}_{\mathrm{VI}}(\pi_2)$
$\pi_1 = \mathrm{M}_{\mathrm{QL}}(\mathrm{M}_{\mathrm{QL}}(\mathrm{R}))$	$V:$ 0.539 $g_t:$ 0.065 $g_1:$ 0.876	$V:$ 0.539 $g_t:$ 0.064 $g_1:$ 0.877		
$\pi_2 = \mathrm{M}_{\mathrm{QL}}(\mathrm{M}_{\mathrm{VI}}(\mathrm{R}))$			$V:$ 0.506 $g_t:$ 0.058 $g_1:$ 0.891	$V:$ 0.506 $g_t:$ 0.058 $g_1:$ 0.893

Table 5.7: Robustness of max-policies against worst-case opponents (6×4 soccer field) with better initial training partners. The explanation of how to read the table is as in Table 5.3.

$0.483 \approx V(\xi_{\mathrm{start}}, \mathrm{MM}_{\mathrm{VI}}(\mathrm{R}), \mathrm{R})$, Table 5.9). However, max-min policies fully exploit suboptimal policies of the opponent with the constraint of staying safe in the sense that max-min methods assume that after the observed non-optimal action the opponent again will act optimally. This prevents the player from being tricked, i. e. that the opponent intentionally behaves in a way to create a misleading conclusion about its policy.[13]

Exploitability of Optimal Max-min Opponents

In addition to the question of exploiting a suboptimal opponent it is interesting to ask whether a max-min policy can be better exploited by a max policy than by a max-min policy. The related theoretical question is whether the max-min policy is already a best response to a max-min policy. In contrast to the case of non-optimal strategies, the answer is positive [186]. In Tables 5.8 and D.11 this – or more appropriate: the software DRPOST – is verified by the fact that both values for $\pi_2 = \mathrm{M}_{\mathrm{VI}}(\pi_1)$ and $\pi_3 = \mathrm{MM}_{\mathrm{VI}}(R) = \mathrm{MM}_{\mathrm{VI}}(\pi_1)$ are equal (necessarily equal to 0).

	$\pi_2 = \mathrm{M}_{\mathrm{VI}}(\pi_1)$	$\pi_3 = \mathrm{MM}_{\mathrm{VI}}(R)$
$\pi_1 = \mathrm{MM}_{\mathrm{VI}}(\mathrm{R})$	$V:$ 0.000 $g_t:$ 0.070 $g_1:$ 0.502	$V:$ 0.000 $g_t:$ 0.071 $g_1:$ 0.503

Table 5.8: Exploitability of optimal max-min opponents (6×4 soccer field). The explanation of how to read the table is as in Table 5.3.

5.2.3 Relating Policies to Humanoid Soccer Characteristics

It is not trivial to get qualitative heuristic results beyond the abstract numerical value of a policy or a pair of policies because there can be differently characterised succesful policies. A standard distinction in human soccer is e. g. between defensive and offensive strategies. A practical way to determine the offensiveness and defensiveness of a grid soccer policy is the following two step method: first, evaluate the policy against a random opponent to test its offensiveness (for a very weak opponent a defensive behaviour does not contribute

[13] Although it is often natural and plausible to assume that the opponent will repeat non-optimal behaviour this can not be relied upon.

to its success) and, second, evaluate the policy against itself. It can be tried to estimate the defensive quality by the reduction of scored goals in comparison to the weak opponent.

Exemplarily, some results of Tables 5.9 and D.12 are evaluated. A look at the first column could lead to some confusion because to stay consistent and keep the page layout the values V are from the point of view of the column policy. To change the viewpoint the negative signs have to be made positive. Then it becomes directly clear that the random policy is by far the worst and that π_3 is strongest against the random policy. Its offensiveness can be seen by the high goal rate per time step g_t and its very high amount of own goals ($1 - 0.016 = 98.4\%$, Table 5.9). The RL analogon π_2 is similarly strong which indicates that the learning phase was sufficiently long.

The max-min optimal strategy π_5 seems to be weaker because of its safety aspects but outperforms the random opponent considerably. This means that it clearly exploits the weaknesses of its opponent although in a safe way which disproves the argument that max-min strategies do not exploit weaker opponents. The defensiveness of the max-min strategy can also be seen in comparison to the max strategy in the second column by noticing that the goals per time step are fewer. Remarkably, the max-min RL method π_4 is stronger against the random opponent than the DP method. This could have two reasons: the first is that the learning is not completed and the safety is not optimal (this is not the case here) and the second reason is that non unique Nash equilibria exist which is verified by the author at least for single states $s \in \mathcal{S}$.[(14)] The Nash equilibrium (optimal) policies are all equally strong against a worst-case opponent (Theorem 2.20), however, they may be and obviously are differently strong against the non-optimal random opponent.

A last highlighted result – again providing an argument against using MDP models for robot soccer – is the evaluation of π_7. Although the max strategy π_7 is trained against π_3 which is the strongest max opponent for the random policy it is relatively weak against the random opponent. This again indicates that the max learning only adapts to that *single* opponent to which it is an approximate best response. In contrast, the quality of max-min solutions is independent from the training partner which can only influence the update order in learning and not the final policy.

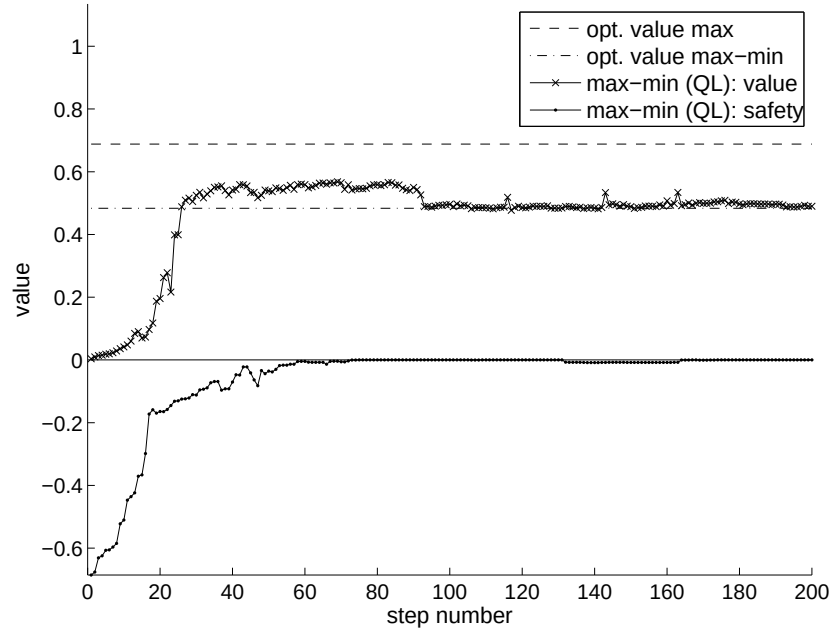
5.2.4 Comparison of DP and RL Techniques

In this subsection a rough impression is to be given of how much more effective DP methods are in comparison to RL methods. The comparison is only depicted for 2P-ZS-MGs (max-min methods) but the result that DP methods are more effective is also expected in classical MDPs (max methods). A novelty is the inclusion of the game theoretic safety measure: the computed policy is not only evaluated against its standard opponent (standard value evaluation) but also against a worst-case opponent (*security level*). Figure 5.6 comprises all details for a 1v1 grid soccer model with MDP typical symmetry reduction (all symmetries which do not exchange the two players are reduced). The DP method converges to a reasonable policy much faster (after 6 steps) than the RL method (after about 24 steps). Concerning the security level, the DP method also needs far fewer steps (17 in comparison to 70 for the RL method) to achieve a value close to 0.

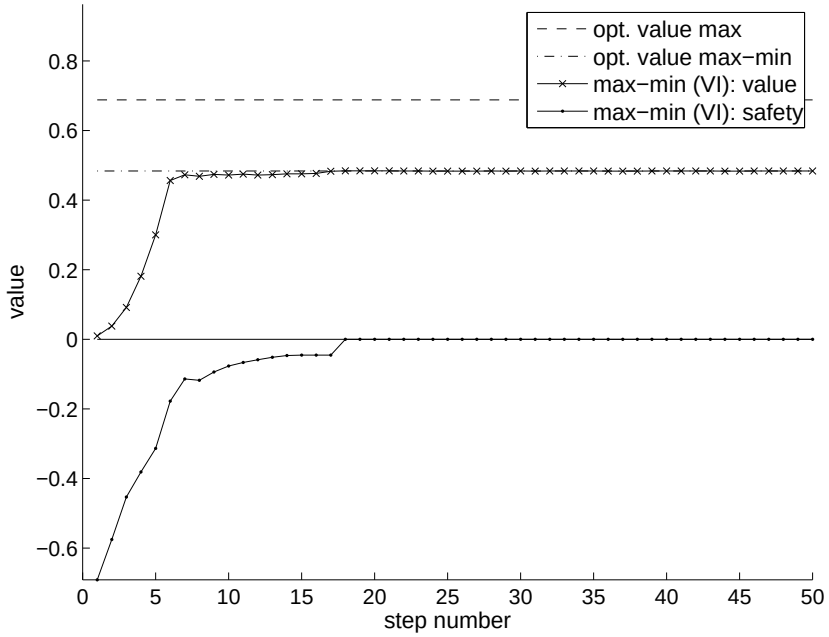
[(14)] The non-uniqueness of matrix game policies of single states $s \in \mathcal{S}$ implies the non-uniqueness of the total policy.

	$\pi_1 = \mathrm{R}$	equal
$\pi_1 = \mathrm{R}$	$V:\ 0.000$ $g_t:\ 0.005$ $g_1:\ 0.525$	$V:\ 0.000$ $g_t:\ 0.005$ $g_1:\ 0.495$
$\pi_2 = \mathrm{M}_{\mathrm{QL}}(\mathrm{R})$	$V:\ -0.682$ $g_t:\ 0.063$ $g_1:\ 0.015$	$V:\ 0.000$ $g_t:\ 0.092$ $g_1:\ 0.499$
$\pi_3 = \mathrm{M}_{\mathrm{VI}}(\mathrm{R})$	$V:\ -0.688$ $g_t:\ 0.064$ $g_1:\ 0.016$	$V:\ 0.000$ $g_t:\ 0.092$ $g_1:\ 0.501$
$\pi_4 = \mathrm{MM}_{\mathrm{QL}}(\mathrm{R})$	$V:\ -0.566$ $g_t:\ 0.053$ $g_1:\ 0.016$	$V:\ 0.000$ $g_t:\ 0.070$ $g_1:\ 0.499$
$\pi_5 = \mathrm{MM}_{\mathrm{VI}}(\mathrm{R})$	$V:\ -0.483$ $g_t:\ 0.046$ $g_1:\ 0.026$	$V:\ 0.000$ $g_t:\ 0.070$ $g_1:\ 0.499$
$\pi_6 = \mathrm{M}_{\mathrm{QL}}(\pi_2)$	$V:\ -0.086$ $g_t:\ 0.011$ $g_1:\ 0.147$	$V:\ 0.000$ $g_t:\ 0.020$ $g_1:\ 0.498$
$\pi_7 = \mathrm{M}_{\mathrm{QL}}(\pi_3)$	$V:\ -0.098$ $g_t:\ 0.012$ $g_1:\ 0.132$	$V:\ 0.000$ $g_t:\ 0.018$ $g_1:\ 0.496$

Table 5.9: Analysis of offensiveness and defensiveness of different policies (6×4 soccer field). The explanation of how to read the table is as in Table 5.3.



(a) RL method.



(b) DP method.

Figure 5.6: Convergence speed of RL and DP techniques measured by value and security evaluation: standard Q-learning versus a Gauss-Seidel DP method ($\gamma = 0.9$, random exploration rate 0.20 for the RL method). The scales for the step numbers (x-axis) are different.

5.2.5 Comparison of Different DP Techniques with Various Parameters

In the following basic studies of different DP techniques with a variety of parameters based on the model of 1v1 multi-player grid soccer are provided. In general, such comparisons should also improve the understandings of RL methods because RL methods offer some additional subtleties (such as choosing a starting state or exploration strategies) which can be avoided in DP methods. Furthermore, in some sense DP methods provide an upper bound for the effectiveness of RL methods (if strict bounds on the quality of the solutions are to be guaranteed) because in DP methods the model is completely known. However, the strength of RL methods is typically to yield a good solution *without guaranteeing* its quality, i. e. that in a typically seldomly occuring state the policy can be arbitrarily bad.

This subsection is divided into different parts: first, the three DP methods with normal update without symmetry reduction, Gauss-Seidel update without symmetry reduction, and Gauss-Seidel update with symmetry reduction are compared for different choices of initial value iterates V_0 and discount factors γ and for the three cases max-min (2P-ZS-MG), max (MDP with opponent being fixed to perform a uniformly random policy), and fixed (Markov process with player and opponent performing a fixed random policy, simply: policy evaluation). Note that the theoretical fourth DP method of normal DP updates with symmetry reduction is the same as without reduction except that the state space is shrunken and therefore each value iteration step is considerably faster. Second, different state sorting strategies for Gauss-Seidel methods are evaluated and an example is worked out that shows that a resorting of the update order can lead to loosing the monotonicity of the Bellman error in MDPs and 2P-ZS-MGs. Third, a special convergence phenomenon is analysed in detail to recover why certain large error improvements occur.

Initial Value Functions V_0 and Discount Factors γ

There are two qualitative standard behaviours of DP and RL methods which are verified in this thesis by means of the grid soccer model. The first behaviour is that the discount factor γ dramatically influences the convergence speed for DP and RL methods: the closer γ is to 1 the more update steps are needed to achieve a prescribed maximal error. The second behaviour is that the closer the initial value function V_0 is to the optimal value function V^* with respect to $\| \, \|_\infty$ the less update steps are needed. Both algorithmic behaviours are direct consequences of the value iteration theorem (Definition 2.35 and below): the first one is due to the fact that the stopping criterion depends on $\frac{1-\gamma}{\gamma}$ and that the contraction rate of the Bellman operator is γ; the second one is due to the fact that $\|V_{k+1} - V_k\|_\infty \leq \|V_{k+1} - V^*\|_\infty + \|V_k - V^*\|_\infty = \|\mathcal{B}_{\mathrm{MG}} V_k - \mathcal{B}_{\mathrm{MG}} V^*\|_\infty + \|V_k - V^*\|_\infty \leq (1+\gamma) \cdot \|V_k - V^*\|_\infty$.

Notes on the Choice of the Discount Factor. It is to be remarked that the conclusion of the first behaviour of DP and RL methods is *not* to use a very small discount factor γ but to use the smallest *reasonable* one. The problem of a too small γ is that e. g. with $\gamma = 0.1$ the 2P-ZS-MG reward after about 10 time steps is discounted to a magnitude below that of numerical errors. One possibility is to determine or to estimate the minimum number of time steps t_0 between obtaining two essential rewards and setting γ in a way that the total discounting γ^{t_0} is in the order of 0.5.[(15)]

A mathematically more beautiful point of view includes *Euler's number* e which can be

[(15)] For $\gamma = 0.9$ this would yield a reasonable time scale of 6 to 7 time steps.

expressed by different limits, e. g.

$$\frac{1}{e} = \lim_{n\to\infty} \left(1 - \frac{1}{n}\right)^n \tag{5.1}$$

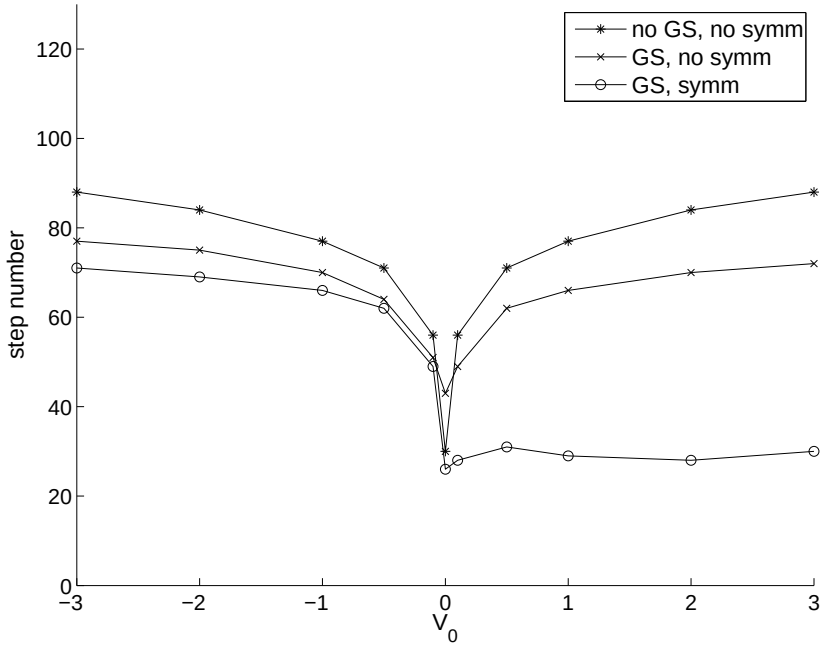
For the special discount factors $\gamma = 1 - \frac{1}{n}$ and n large enough this implies that the n-step discounting is close to $\frac{1}{e} \approx 0.368$. If a discrete 2P-ZS-MG is constructed by discretisation of an underlying continuous model, then doubling the discretisation accuracy will typically lead to a doubling of the needed discrete time steps to describe the same process. Hence to maintain expressiveness of the numerical results a sensible γ would also be double as close to 1.

Notes on the Magnitudes of the Value Function. The values for V_0 are chosen by the following criteria: 3 is larger than the maximum optimal value, -3 is lower than the minimal one. ± 1 is in the magnitude of high or low values while ± 0.1 is a small positive or negative estimation. 0 is the mean value of the value function for the max-min case because a 2P-ZS-MG μ-isomorphism with $\mu = -1$ exists. For the max case the mean value against a random opponent is 0.688 for $\gamma = 0.9$, 0.136 for $\gamma = 0.75$, or numerically zero ($\approx 1 \cdot 10^{-7}$) for $\gamma = 0.1$ which shows that this discount factor is far too small for the grid soccer problem. The relatively high positive values of start states for the first two discount factors show that it is (of course) much better to use an optimised strategy as to act randomly.

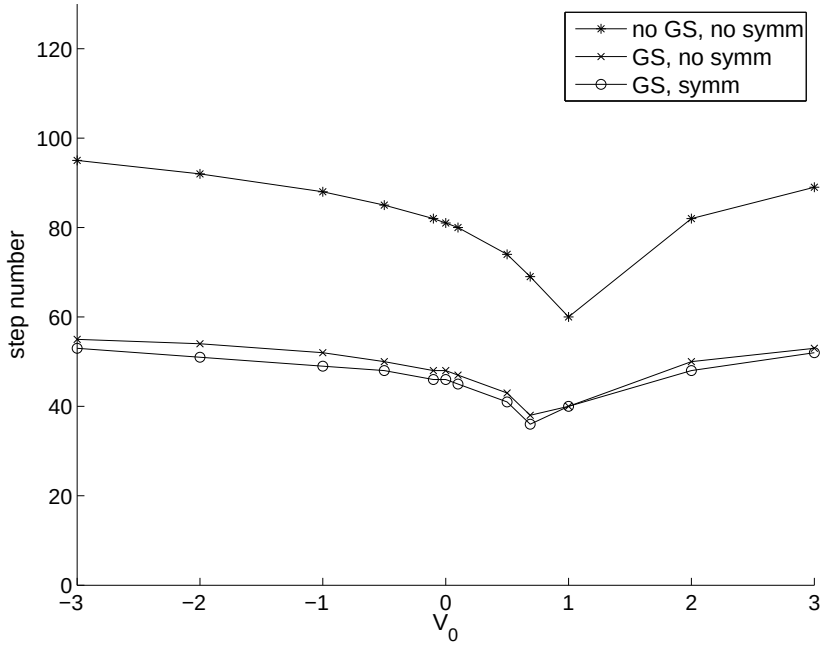
Notes on the Iteration Steps. Figures 5.7 and 5.8 are intended to give a quantitative feeling about how many iteration steps different DP methods need for convergence to a certain error ($\varepsilon = 1 \cdot 10^{-3}$) for max-min DP methods and max DP methods (with a randomly acting opponent). The effects of different values for γ and V_0 and of using standard and Gauss-Seidel type updates as well as using symmetry reduction are illustrated.[(16)] The analogue results for iterative policy evaluation, i. e. that both players of the 2P-ZS-MG are acting according to a fixed – here: random – policy are omitted at this place (see Figure D.1) because they give a qualitatively similar picture as the max methods do. The Gauss-Seidel methods work more efficiently than the standard updates in the max case but there is no essential difference between Gauss-Seidel methods with and without symmetry reduction except that each iteration consists of a lower number of updates. Astonishingly, for max-min methods the general view changes qualitatively: for $V_0 \geq 0$ the symmetry reduction decreases the number of iterations considerably. The effect is larger the more iterations are needed by the other two methods which leads to a *particular insensitivity* of this method to the values of $V_0 \geq 0$: for a Gauss-Seidel method without symmetry reduction the number of steps k for $V_0 \in [0, 3]$ ranges from 43 to 72 whereas the range of a Gauss-Seidel method with symmetry reduction is only from 26 to 30. However, for negative initialised V_0 this insensitivity is not observable but at least the symmetry reduction leads to the smallest number of iterates in every single case. Two things are essential: first, the occurance of the insensitivity effect and, second, the unexpected dependency on the positivity of V_0. In RL methods a well known phenomenon exists that an initial guess of V_0 which is too positive encourages exploration of unexplored states simply because more realistic estimates of explored states reveal unattractivity of these states [202]. However, for DP methods an analogon does not exist. Perhaps the effect can be explained by the solution of the matrix games: if $V_0 > 0$ and only a few entries of a matrix have changed to more realistic lower ones the worst-case opponent forces the corresponding actions and the value decreases appropriately after a few iterations. If, however, $V_0 < 0$ than nearly

[(16)] Additional tables and figures on the numerical results can be found in Appendix D.

all entries of a matrix, i. e. all values of successor states, have to be estimated in a more realistic way (more positive) before the value of the matrix game is significantly influenced because by the same argument as above a few estimates dominate which are too negative.



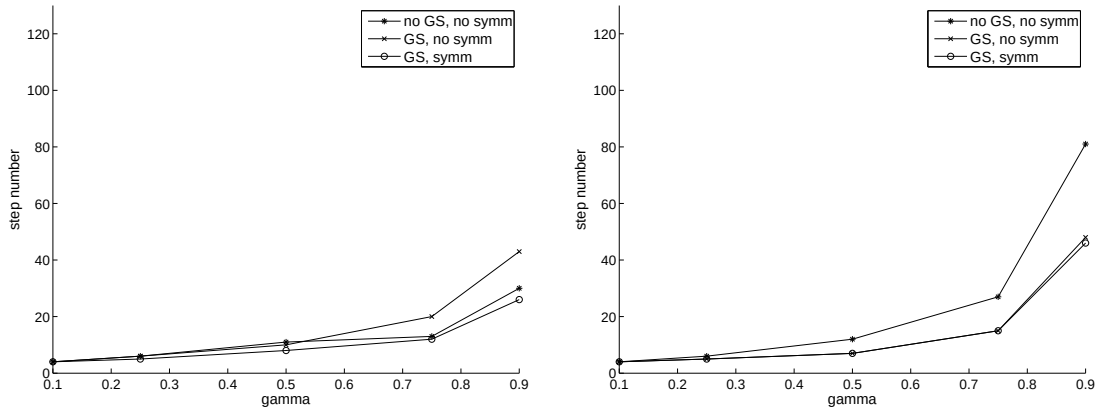
(a) Max-min.



(b) Max.

Figure 5.7: Comparison of different Gauss-Seidel types with or without symmetry reduction: number of iteration steps over initial values of the initial value function V_0 for a 1v1 multi-player grid soccer model and a (a) max-min method, (b) max method without sorting strategy ($\gamma = 0.9$, stopping criterion precision $\varepsilon = 1 \cdot 10^{-3}$ (Corollary 4.3)).

In contrast to the exciting results of varying V_0, the results for varying the discount factor γ presented in Figure 5.8 are completely unspectacular. The result here is valid for max-min, max, and fixed strategy methods and is as above for the max method, namely that Gauss-Seidel type updates work better than standard ones and the symmetry reduction does not have a major effect on the number of iterations.



(a) Max-min. (b) Max.

Figure 5.8: Comparison of different Gauss-Seidel types with or without symmetry reduction: number of iteration steps over the discount factor γ for a 1v1 multi-player grid soccer model and a (a) max-min method, (b) max method without sorting strategy ($V_0 = 0$, stopping criterion precision $\varepsilon = 1 \cdot 10^{-3}$ (Corollary 4.3)).

Different Sorting Strategies for Gauss-Seidel Methods

Figure 5.9 shows the convergence speed of the following different sorting strategies: standard updates (no sorting), updating by maximal Bellman error after each iteration, updating by randomly rearranged state order in each iteration (random), and keeping the randomly rearranged state order of the first iteration (random fixed). Some of the parameters explored above are fixed: $V_0 = 0$ and $\gamma = 0.9$. There are three surprising observations in this figure: First, the "Max-min Convergence Boosting Phenomenon" which means that occasionally a very large error reduction of nearly one order of magnitude occurs within mostly one but at most two iteration steps. This phenomenon has been only observed for max-min methods and looks such important that it is treated in an extra subsection of Section 5.2.5. Second, the update order by highest Bellman error is the worst of all methods for nearly all iteration steps *but* for this update order the large decisive second "convergence boost" occurs first for the max-min method. It is not clear whether this is a coincidence or not because the first smaller "convergence boost" appears later than for the two other methods. The bad performance of Bellman error update order is interesting because the argument to firstly update states with the highest error is directly plausible. Third, the DP methods with Gauss-Seidel type updates without equally sorted states in each iteration step (e. g. Bellman error and random sorting) sometimes show a small error increase. This is *not* due to numerical errors which are smaller than $1 \cdot 10^{-5}$ and thus undetectable in the figure, but to the fact that these update methods do not make use of all estimated values of all value iterates, see Example 5.2. Numerical studies indicate that the true error to the unknown optimal value function is reduced but that the Bellman error does not reflect that fact. Therefore, it is stressed that Bellman errors do not reflect true

convergence but only convergence of a guarantee for convergence. This also gives reasons for the bad performance of sorting by the Bellman error.

The following example is very simple but much can be learnt from it:

5.2 Example (Monotonicity of Bellman Error for some "Gauss-Seidel" Methods) Be $\mathcal{M} = (\mathcal{D}, \mathcal{S}, \mathcal{SA}, T, R)$ an MDP with decision epoch $\mathcal{D} = \mathbb{N}_0$, state space $\mathcal{S} = \{s_1, s_2\}$ (only two states), $\mathcal{A}(s_1) = \mathcal{A}(s_2) = \{a_1\}$ (only one action in every state), $T(s_1, a_1, s_2) = 1$ and $T(s_2, a_1, s_2) = 1$ (probability 1 to go to state s_2 and stay there), and $R(s_1, a_1) = 0$ and $R(s_2, a_1) = 1$.
In fact, this example is a discrete-space discrete-time dynamical system since only one policy exists (probability 1 for the only action in each state) which is concurrently the optimal policy. Furthermore, it represents the simplest example with one recurrent and one transient state (in terms of dynamical systems). Nevertheless, it gives useful intuition for clustering methods interpreting one state as a cluster of other states and already an example that *in MDPs and 2P-ZS-MGs the Bellman error of Gauss-Seidel methods with reorganising the update order is not monotone.* This is in contrast to the Jacobi update type and to Gauss-Seidel type updates with fixed update order for which the Bellman error decreases at least with the discount factor γ ([168] gives a proof for MDPs). The above statement can be seen by the following table in which the Jacobi and a Gauss-Seidel method with the noted update order is performed for a discount factor $\gamma = 0.9$. The value iterates V_k are represented by the vector $(V_k(s_1), V_k(s_2))$

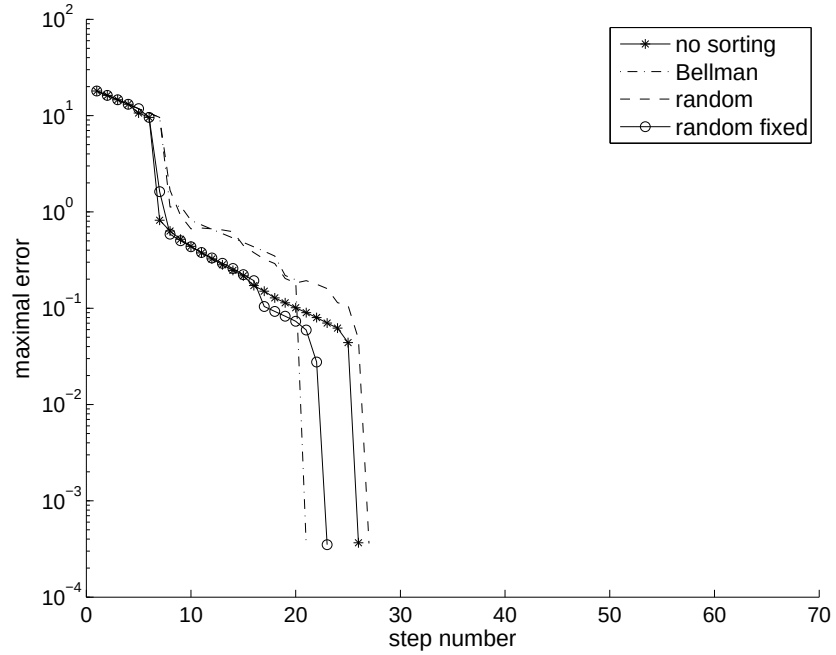
k	V_k (Jacobi)	$\|V_i - V_{i-1}\|_\infty$	V_k (Gauss-Seidel)	update order	$\|V_i - V_{i-1}\|_\infty$
0	$(0, 0)$		$(0, 0)$	s_1, s_2	
1	$(0, 1)$	1	$(0, 1)$	s_1, s_2	1
2	$(0.9, 1.9)$	0.9	$(0.9, 1.9)$	s_1, s_2	0.9
3	$(1.71, 2.71)$	$0.81 = 0.9^2$	$(2.439, 2.71)$	s_2, s_1	$1.539 > 0.9$

The problem why the Bellman error is not monotone decreasing by γ is that the updates do not correspond to the Gauss-Seidel method any longer in the strict sense since for computing $V_3(s_1)$ the value $V_3(s_2)$ is used whereas for $V_2(s_1)$ the value of $V_1(s_2)$ is made use of. Thus, $V_2(s_2)$ is not plugged in the two computations which is different for the Jacobi method or for any method with fixed update order.

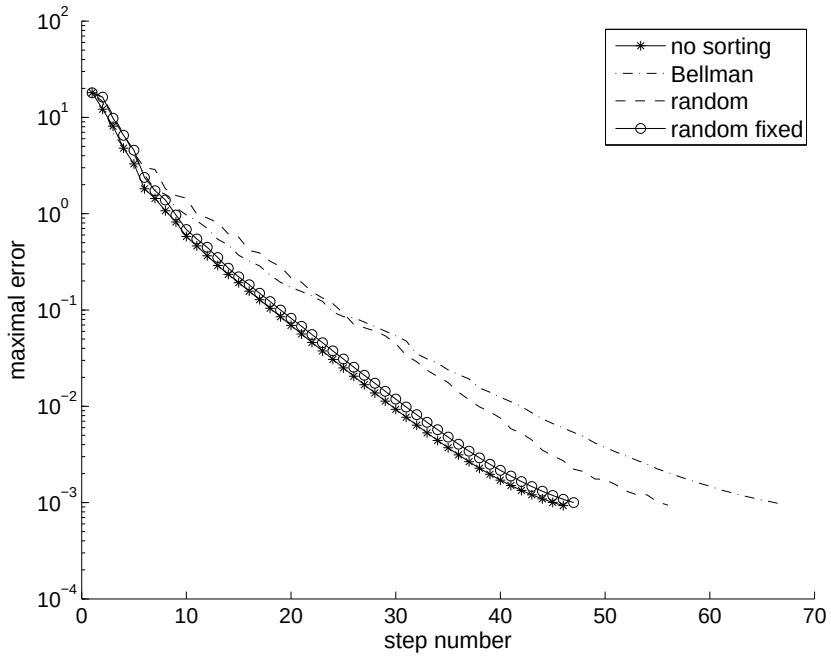
The Max-min Convergence Boosting Phenomenon

At a very first glance, it would not be surprising if a max-min value iteration would take more steps than a max or fixed strategy value iteration for the reason that 2P-ZS-MGs are more complex than MDPs or Markov chains. However, the fixed policy method needs more steps than the max-min method, whereas the max method needs clearly the most number of iteration steps.

A heuristic argument is that of information diffusion: the shorter the longest shortest path is from any state s_i to s_j on an appropriate graph, e. g. induced by the transition matrix, the faster information about the value iterate $V_k(s_i)$ is propagated to the state s_j in a future iterate. Because the success of value iteration is measured by $\| \|_\infty$ it can give bad



(a) Max-min.



(b) Max.

Figure 5.9: Maximal DP error (logarithmic scale) over the number of iteration steps for a Gauss-Seidel type update with symmetry reduction, a 1v1 multi-player grid soccer model, and a (a) max-min method, (b) max method by means of standard updates (no sorting), Bellman error estimation (Bellman), randomly rearranged state order in each iteration (random), and keeping the randomly rearranged state order of the first iteration (random fixed) ($V_0 = 0$, $\gamma = 0.9$, stopping criterion precision $\varepsilon = 1 \cdot 10^{-3}$ (Corollary 4.3)).

performance if only a few states are late informed about significant changes. If only the number of connections by the transition matrix without a detailed shortest path analysis is considered this will give reason why the fixed strategy method needs less iterations than the max method because the first includes two completely random policies and the second includes one random and one (nearly) deterministic policy. Applying the same argument, the max-min method should lie in between the two other methods because only sensible actions are performed with non-zero probability by each of the two players but amongst these sensible actions a reasonable diversification has to be performed to minimise the risk of being exploited. However, because of extraordinary convergence steps the max-min method seems to be the fastest.

A less heuristic argument is based on iterative linear solvers. The Bellman equation for fixed policies is equivalent to solving a linear equation but also the non-linear versions can a posteriori be reformulated in a linear way when the optimal policies are known which fulfill the max or max-min equation. Thus, the convergence speed can be analysed by the same methodology standardly employed for iterative linear solvers (see Appendix B). It should be remarked that there is a small difference between Bellman equation Jacobi and Gauss-Seidel update and the classical versions for iterative linear solvers. All in all, the analysis does not reveal novelties: the $\| \ \|_\infty$ for the update matrix nearly always equals exactly the discount factor γ and even the spectral radius is always close to it. That implies that even with a different norm – which can not be chosen freely for the convergence guarantee – no significantly better convergence rates can be explained. The conclusion of this paragraph is that the worst-case convergence rate is not a good measure for the real convergence rate in the practical example of multi-player grid soccer.

A next idea *why* the "Max-min Convergence Boosting Phenomenon" might occur is the existence of 2P-ZS-MG μ-isomorphisms with $\mu = -1$ (player exchanging symmetries). It was mentioned in Section 3.2 that this qualitatively new kind of symmetry is special to 2P-ZS-MGs and can not occur in MDPs. Hence, in Figure 5.10 the same algorithmic convergence as in Figure 5.9, a) is shown with the exception that the max-min method without reduction of the player exchanging symmetry is considered (all other symmetries are reduced). The result is that the convergence boosting also takes place similarly. Also, the possibility that the Gauss-Seidel update could be responsible is excluded by observing the same phenomenon with non Gauss-Seidel type updates.

Nevertheless, Figure 5.10 reveals a new idea: without the player exchanging symmetry the steps with convergence boosts seem to have a clearer periodical structure. This is valid at least for the updates with fixed update order ("no sorting" and "random fixed" in the Figure) and the other update types should be neglegted because of Example 5.2. The periodicity is more clearly illustrated in Figure 5.11, b) in which the stepwise convergence rate is depicted. The rough periodicity of $6 - 7$ steps for the peaks is in concordance with the expected duration of scoring a goal after a restart of the game. It is speculated that this inherent "periodicity" of the model is responsible for the periodicity of the convergence error peaks. This would also fit to the heuristic argument of information diffusion above. Furthermore, the differences of the Bellman error and the true error indicate that the Bellman error often decreases later than the true error.[(17)]

[(17)] The true error is computed by the difference to the optimal value function. The total reduction of the Bellman error can be larger than the total reduction of the true error since the initial error for the Bellman estimate is largely above the true error.

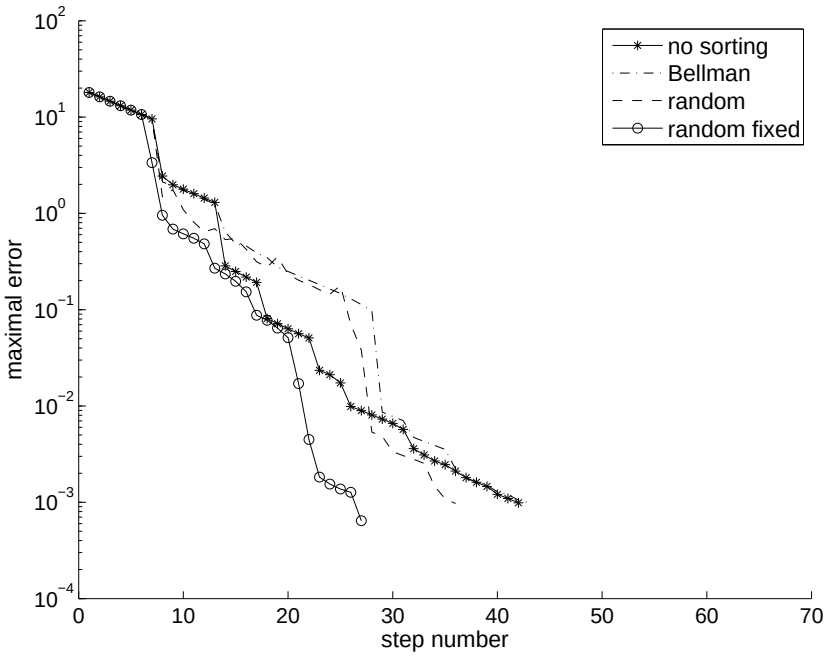


Figure 5.10: Maximal DP error (logarithmic scale) over the number of iteration steps, all details are as in Figure 5.9, only the player exchanging symmetry is not reduced.

5.2.6 Comparison of Standard Methods and SL Techniques

It is pointed out in Lemma 4.2 that convergence guarantees can not be given if the approximation error introduced by the SL techniques is close to the magnitude of the desired error of the value function. Nevertheless, SL techniques have been successfully combined with RL methods (Section 4.3) and in this spirit the practical effort of the following results is to be understood.

Especially for single player robot soccer [97] introduces a number of features. These seem to be highly specialised to the model which is quite similar to the 1v1 multi-player grid soccer of Section 5.1 because both models are motivated by [112]. The features are case based and extract the information in a way such as "*if* the attacker is not closer to the defenders goal than the defender *and if* the defender is close to the attacker then store some position information of the defender in relation to its goal and all possible relative positions between the attacker and defender" and so on for all four possible if-combinations.

In contrast to the probably very laborious work to design such features, the approach used in the present work is to approximate all occuring intermediate state value functions $V(s)$ during an RL procedure by the following very simple features which are already designed for their application in multi-player models:

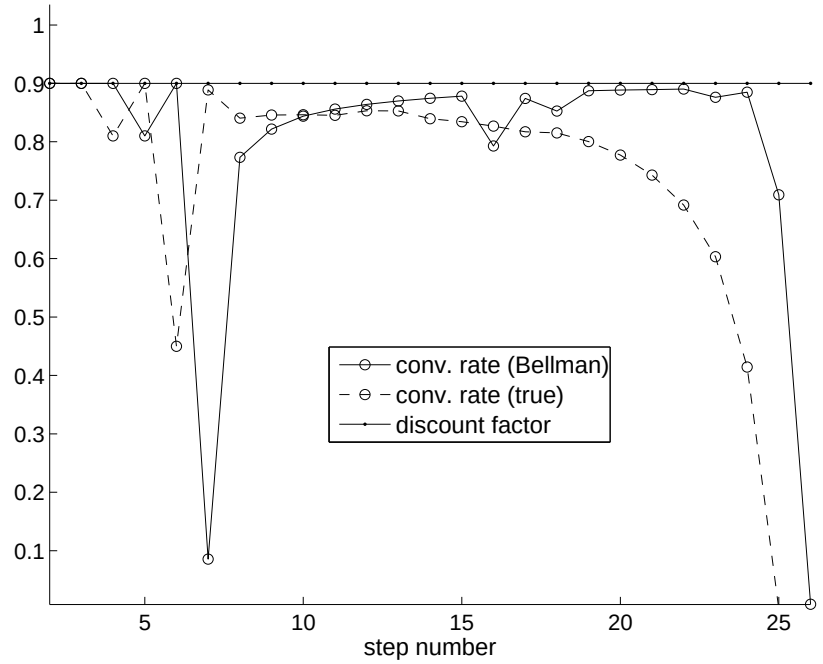
1.) the number of robots of each team in the cell of the ball,

2.) the minimal distance of each team (minimum over all team members) to the ball,

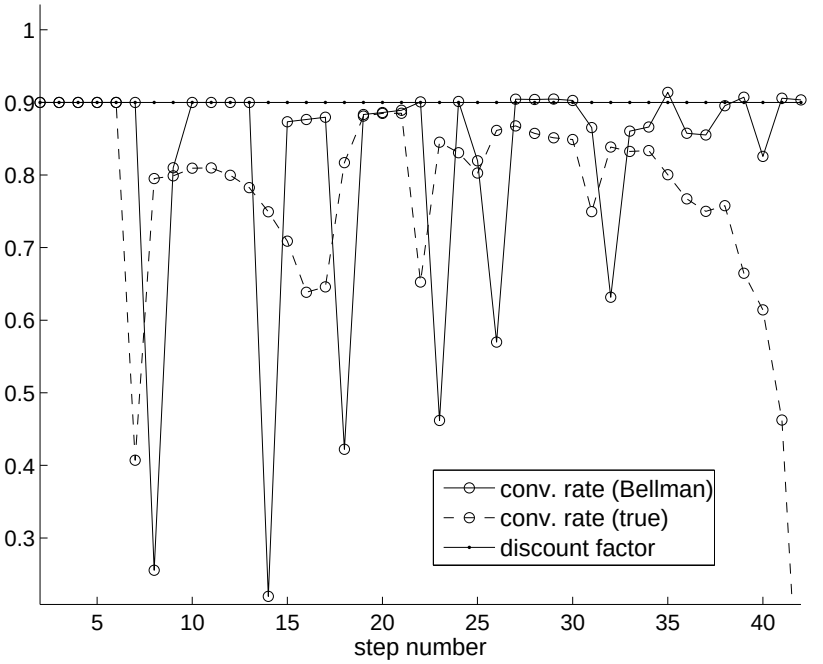
3.) the minimal distance of the ball to each goal region, and

4.) the number of robots of each team in the half of the first team.

Distances are measured by squared Euclidean distance and all features are restricted by a maximum value, e. g. 3 for a small grid. If the distances to the goal regions are not restricted one of the two would suffice. Table 5.10 shows results of a policy based on state



(a) Symmetry reduction 1.



(b) Symmetry reduction 2.

Figure 5.11: Convergence rate of a max-min DP method by maximal DP error (Bellman) and by the true error over the number of iteration steps for a Gauss-Seidel type update and a 1v1 multi-player grid soccer model. In (a) all symmetries are reduced, in (b) all except the player exchanging symmetry are reduced, i. e. (a) corresponds to the "no sorting"-line of Figure 5.9 (a), and (b) corresponds to that line of Figure 5.10. The convergence rate can be a little larger than γ for later iterates because the Bellman error includes the results of Lemma 2.36.

value functions represented by feature approximation architecture (92 degrees of freedom instead of the 576 needed for the lookup table representation). The max-min solution of the 1v1 grid soccer model with respect to the exploitation of the random opponent (in reference to the discussion about non-unique Nash equilibria) are reasonably good but not as good as that of Table 5.9. The security level against a worst-case opponent trained with the same restrictive architecture is perfectly optimal, i. e. equal to 0, because both players are restricted to the same policy space. Nevertheless, something is lost by approximating the value function only by features which could also be interpreted as building "wrong equivalence classes" by means of heuristics. This becomes obvious when looking at the high outperformance of a worst-case opponent which is *not* restricted to the feature representation of the value function: the separated right column reveals that the policy π_1 against this opponent is quite bad.

The conclusion of the above is that features can yield reasonable results even if they are very simple and also that the security level against an opponent of the same feature architecture is reasonable. However, the performance can significantly degrade against more powerful approximation architectures. This makes the knowledge about which feature architecture is used by the opponent nearly as useful as to know its policy directly.

	$\pi_2 = \mathrm{R}$	$\pi_3 = \pi_1$	$\pi_4 = \mathrm{M}_{\mathrm{QL}}(\pi_1)$	$\pi_5 = \mathrm{M}_{\mathrm{VI}}(\pi_1)$
$\pi_1 = \mathrm{MM}_{\mathrm{QL}}(\mathrm{R})$	$V:\ -0.325$ $g_t:\ 0.033$ $g_1:\ 0.054$	$V:\ 0.000$ $g_t:\ 0.068$ $g_1:\ 0.502$	$V:\ 0.000$ $g_t:\ 0.068$ $g_1:\ 0.500$	$V:\ 0.423$ $g_t:\ 0.042$ $g_1:\ 0.955$

Table 5.10: Evaluation of max-min policies for a 1v1 grid soccer on a 6×4 field which are computed with a feature based value function. Only the separated right-most column strategy π_5 is computed without features.

5.2.7 Towards Multi-Player Robot Soccer: 2v2 Grid Soccer

In this section effects of large state spaces obtained by a fine grid discretisation of 1v1 grid soccer and large state spaces induced by multi-player especially 2v2 grid soccer on a coarse grid are to be discussed. Tables 5.11, and 5.12 show the number of state and state-action spaces as well as the number of value iterations needed for convergence. From a heuristic point of view the discretisation to a higher grid size can be thought to be easier, however, this is not reflected in the number of value iterations. However, the computational time for the 2v2 case is much higher per iteration because the size of the matrix games is growing quadratically in the exponentially growing action space of the player (since the action space of the opponent is growing exactly as fast). A further expectation of the author is that there should only be a small amount of extra information obtained by the solution of a finer resolution of the grid.[(18)] Nevertheless, it should be expected that an initialisation of a finer grid with an adapted value function of a coarser grid should lead to a remarkable reduction of needed iteration steps because the fine and coarse grid models possess strong similarities.

In contrast, for a 2v2 grid soccer model it should be much more difficult to transfer knowledge from the 1v1 grid soccer model. The main aspect is that the structure of the model

[(18)] Note that the different grid size models are different models and can not be interpreted as two different discretisations of a common underlying continuous model.

Size	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{SAO}\rvert$	Results
small (6x4)	282	4893	26 0.00037
medium (12x8)	4584	96288	33 0.00039
large (24x16)	73632	1690578	47 0.00041

Table 5.11: Comparison of symmetry reduced 1v1 grid soccer with $\gamma = 0.9$ (different soccer field sizes) by problem size and necessary DP iterations for achieving a stopping criterion precision of $\varepsilon = 1 \cdot 10^{-3}$ (Corollary 4.3). The achieved precision is also noted.

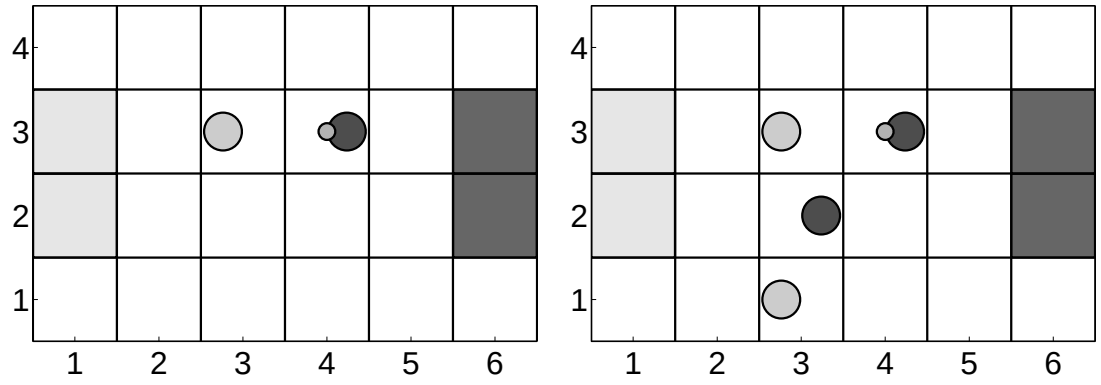
Size	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{SAO}\rvert$	Results
small (6x4)	82944	27724780	19 0.00075

Table 5.12: Analogon to Table 5.11 for 2v2 grid soccer for a 6×4 soccer field.

drastically changes: the new options of performing passes and to coordinate behaviour with a second team member can lead to a completely different kind of optimal behaviour. Since it is a little tedious and will not give many insights to analyse the *whole* policy or value function the author tried to identify a single situation which is somehow comparable but in which differences of a 1v1 and a 2v2 grid soccer optimal policy become obvious.

The state in Figure 5.12 a) for the 1v1 grid soccer is extended to 2v2 soccer by adding a tactically not so well positioned second opponent and a tactically well placed second player ready for a pass to the field. In the 1v1 case the optimal policy is to go left with probability 1 because the opponent player can only block the ball if he can be in the same cell after one move as the ball-possessing player, i. e. the opponent player has to stand in his cell. Also moving above or below does not make sense for the player because the opponent player will then in the next turn definitely meet him or else a time consuming phase of several side turns would start and decrease the time discounted long-term reward. In the 2v2 case the situation with the added team members the situation looks different: the optimal policy for Team 1 is to let the member without ball move to the left and the other pass to it with a probability of roughly $\frac{2}{3}$ – although there is a real chance that a miskick will occur[19] – and to perform with the remaining probability of $\frac{1}{3}$ the same movement for the team member without ball but a move downwards without a kick for the ball-possessing one. The tactical advantage of this position is that there is a new chance for a kick without the opponent team being able to reach the ball-possessing team member. The opponent team has a clearer strategy: the opponent in the middle at coordinates $(3, 3)$ has to stand and the second opponent has to stand with a probability of 93% and to go upwards with a probability of 7%. The key aspect is here for both opponents to stay in a range to be able to be a mistarget of a possible pass.

[19] Without a miskick this action would have probability 1 because then the goal scoring would be secure after the pass.



(a) 1v1 game (b) 2v2 game

Figure 5.12: Case study for a situation of 1v1 and a similar situation in 2v2 soccer. Dark grey indicates the defended goal region and the players of the first team, light grey those of the second team and the ball is sketched by the small grey circle.

5.3 A New Algorithm: MaG-Clus-VI

In the following a new algorithm which combines local and global update steps by means of clustering methods is introduced. The idea of structuring the state space with the use of clustering algorithms is common with the graph clustering by topology approach of [130]. Nevertheless, there are two key differences: firstly, the following clustering algorithm takes into account the *dynamics* by clustering by the transition probabilities and not simply by values of the value function or by some state space topology, and secondly no macro actions on clusters are considered. This makes the theoretical proof of convergence very simple and a loss of optimality with respect to the original model can be avoided. An optimal solution will always be achieved if every state is updated infinitely many times ([19] for MDPs) which can be guaranteed by performing infinitely many global steps or by arranging the local updates in a way that after all local updates thought of as a "big local step" at least every state is updated once. This criterion is met by the algorithm below called Markov Game Value Iteration with Clustering (MaG-Clus-VI) because the union of all partitions forms the state space.

5.3 Algorithm (Markov Game Value Iteration with Clustering (MaG-Clus-VI))
Given a 2P-ZS-MG $\mathcal{M} = (\mathcal{D}, \mathcal{S}, \mathcal{SAO}, T, R)$ with $\mathcal{D} = \mathbb{N}_0$ the MaG-Clus-VI algorithm is defined by:

1.) Global steps:

Perform $n_{\text{glo}} \in \mathbb{N}_0$ steps of standard value iteration. The result is an estimation V of the value function as well as a Nash equilibrium policy π with respect to V.

2.) Local steps:

a) Determine the transition probability matrix with entries for all state pairs $(s_i, s_j) \in \mathcal{S} \times \mathcal{S}$ by the 2P-ZS-MG transition probabilities and the total policy π of both players.

b) Use any clustering method, especially for obtaining balanced partitions, to determine the k almost invariant clusters $C_i \subseteq \mathcal{S}$, $i = 1, \ldots, k$.

c) For $i = 1, \ldots, k$: perform for C_i standard value iteration updates restricted to C_i

until convergence to accuracy $\varepsilon = \gamma^{m_i} \cdot \varepsilon_{\text{prev}}$, $m_i \in \mathbb{N}$ is achieved, whereat $\varepsilon_{\text{prev}}$ is the Bellman error of the last global step of 1.) if $n_{\text{glo}} > 0$ or the estimation of the extra convergence check in 3.) if $n_{\text{glo}} = 0$.

3.) Go to 1 until some convergence criterion is achieved. If $n_{\text{glo}} = 0$ an extra step of global value iteration can be performed without influencing the next estimate of V otherwise during the global steps the approximation error can be estimated without extra computational effort.

One strength of the above algorithm is that in the local steps the propagation of larger updates is restricted to only a part of the state space and on that part fully exploited. Another key point is that the algorithm can have a various number of local update steps on C_i and C_j which adapts the update steps to the local properties of the model. Finally, it should be advantageous to use almost invariant sets because this minimises the influence of value updates on adjacent partitions. Furthermore, the almost invariant sets are those which trap a typical Q-learning trajectory which follows an optimal policy for a relatively long time. All in all, the MaG-Clus-VI-algorithm combines the idea of global updates with the idea of local exploration of Q-learning. On the one hand it is global and on the other hand hierarchically structured without loosing the guarantee to obtain an optimal policy of the *non*-hierarchical model.

Some numerical results are obtained with the 2P-ZS-MG of 1v1 grid soccer: with the number of partitions being equal to 4 in each step, $n_{\text{glo}} = 7$, and $m_i = 3$ for all steps and all i the algorithm needs about 39.2 steps for obtaining a prescribed accuracy of $\varepsilon = 1 \cdot 10^{-3}$. If using the partition created by the optimal policy from the first step – which assumes that the optimal policy problem is already solved before – this method takes about 39.8 steps which shows that the adaptivity may be more advantageous than the knowledge of the clustering by the optimal policy. In comparison to the 42 steps of the standard Gauss-Seidel method this is about 5% less, however, for random clustering 49.5 iteration steps are needed. Thus, the comparison with "knowledge free" clustering yields a reduction of about 20% by use of almost invariant sets.

5.4 From Grid Soccer to Robot Soccer: Practical Issues

There is an essential variety of practical issues of transferring the numerical results obtained in MATLAB by DRPOST to real robots, e. g. to AIBO ERS-7 type robot dogs. Some of the key aspects are the use of lower level behaviours which are considered to be elementary actions in the 2P-ZS-MG model and the extraction of visual information especially the self and opponent localisation task. Localistion of the robot and all other robots is essential for determining the state $s \in \mathcal{S}$ and therefore for being able to apply a state-dependent policy.

Since it would go far beyond the scope of a single PhD thesis to design a software architecture from scratch which handles the control of single joints of the robots reasonably well, basic behaviours such as walking and kicking, higher level behaviours, and finally a policy, communication between the robots via WLAN, analysis of (noisy) visual information of a moving camera, and so on, the author decided to resort to the public available software of the *German Team.*[20] A new version of this software is typically published some time after the most recent world competition of robot soccer called RoboCup. A complete *team*

[20] Web site (30.11.2007): `http://www.germanteam.org/tiki-index.php`.

report of 2004 exists, available only online at the German Team web site, which describes all methods and features of the software in some detail.

5.4.1 Lower Level behaviours

Lower level behaviours are provided by the German Team software in a sufficient number. A large number of different walking types already exists with or without the ball being in possession of the robot, an even larger number of different kicks applicable from diverse relative positions and targeting different locations, and a variety of head moving behaviours also exists which is essential for the following localisation task.

The task at hand is to construct action schemes for the robots which correspond as closely as possible to the basic actions $a_i \in \mathcal{A}(s)$ of the multi-player grid soccer model. Some practical experiments show that this task is manageable by manually experimenting in some standard situations of robot soccer.

5.4.2 Image Processing and Localisation

The issues concerning image processing and localisation are trickier than using the lower level behaviours to create the basic actions $a_i \in \mathcal{A}(s)$ of the 2P-ZS-MG. Localistion of the robot and all other robots is essential for determining the state $s \in \mathcal{S}$.

Image processing deals with the problem of extracting information from a series of camera pictures. The basic problem in robot soccer is to match shapes and colors of predefined object to pixels of a picture. The objects as well as the camera can be (not completely but considerably) positioned freely in a three dimensional space such that some objects can be partly hidden or be out of the visible range. In addition to the basic problem noise distorts the picture and the analysis must be performed in real time on the robot itself.

[101] gives an overview and further references of newer and possible developments of image analysis with a focus on robot soccer. The main idea is shortly presented: A set of hypotheses of positions for each object is generated and Kalman filters are applied to predict the moving of each single hypothesis. Then, unlikely hypotheses are removed whereas new ones can be added if corresponding objects are detected in the current camera image.

Self Localisation. Some aspects of self localisation in robot soccer with the AIBO ERS-7 robots are shortly discussed to provide insights for the opponent localisation. The self localisation typically works as follows: in a picture some fixed objects of known size and position (goals, landmarks, lines of soccer field) are detected and by the size on the picture the distance is estimated. This is necessary since only one camera is available to the robot and no stereo three dimensional vision is possible as it is e. g. for humans. To calculate the position in a global coordinate system of the soccer field the position and direction of the camera (in principle all joints of the robots) have to be known.

Summarising, two possible sources of errors are present: first, the errors of reading the sensor information of the joints (error of camera position) and, second, the error of color noise (pixel errors) and misclassified objects (errors of object recognition). It is no issue to locate other team members because it is allowed to communicate via WLAN within a team and to spread the information of the robot's own position.

Opponent Localisation. Since opponent localisation is as essential as self localisation

for the determination of the state $s \in \mathcal{S}$ the additional barriers to obtain a good estimation are shortly outlined. All issues of error sources mentioned in self localisation are present and additionally it is essential that the opponents are moving targets and that not all of them can be seen by each robot at the same time. The reason why this causes problems is that the distance can not be estimated by the size of the robots' trikots because they have a complicated non-convex shape.[21] If it is assumed that only the direction but not the distance can be determined, a natural approach will be to look at the same opponent robot by different team robots and intersecting the corresponding lines. This could work for one opponent robot but even if two opponent robots are present at the same time intersections at locations where no robot stands can occur.

5.5 Other Applications

Many other examples than robot soccer exist in which RL methods are succesfully applied; an overview can be found in [88, 202]. However, two of the main areas are intertwined by robot soccer: game playing and robotics. For example, the lower level of motion control can be considered a typical robotic application and the higher level of strategic planning is strongly related to game playing. This makes robot soccer a challenging and especially interesting subject of research.

Games

The major part of RL literature deals with MDPs which do not include game playing or model only a fixed policy of the second agent. Sometimes, even robotic control problems can be appropriately modeled by 2P-ZS-MGs, e. g. if the aim is to obtain policies being robust to errors of sensors or motors (MDP with "uncertainty generator" as a competitive player [147]). Nevertheless, there are examples that RL methods (Section 2.4) are utilised to solve 2P-ZS-MGs [112, 97].

2P-ZS-MGs include a broad class of games e. g. nearly all two-player board games and two team sport games. Two famous examples of board games are checkers and backgammon in both of which the machine learning can be performed by *self-play*, i. e. a computer plays against itself. Self-play can be seen as a special asynchronous method in which the policy of the first agent is a best response to a time-depending policy of the second agent and vice versa. Hence, the standard assumption for convergence of RL methods need not be fulfilled because instead of a max-min a max policy against the current policy of the second agent which is equal to the own policy is determined.

A very early succesful application was the checkers[22] playing system of Samuel [179, 180] although Samuel did not use the standard RL approach with rewards. Instead, he backed up a kind of value function to estimate the use of board positions. In [202] it is discussed how to relate Samuel's checkers to current RL methods.

A second example which led to remarkable success was the backgammon policy developed by Tesauro's software TD-Gammon [205, 206, 202]. A backpropagation neural network approach with three layers to approximate the probability of winning the game is used. The most successful version of backgammon programs combines the state information

[21] [101] suggests the alternative to estimate the contact point of the robot to the ground. However, no results of the obtained accuracy of detecting multiple moving robots are stated.
[22] Checkers is called "Dame" in Germany.

with features designed by humans. In this way, human insights can be integrated without having a negative influence on the performance. The trade-off between exploitation and exploration is neglected but due to the stochasticity of rolling the dice even a greedy policy execution seems to lead to enough exploration.

Robotics

Robotic tasks often have an inherent continuous nature (states, actions, time) and are demanding because the decision making is disturbed by noise and error, information often has to be extracted from sensor information (actuator status, image processing, speech analysis), and limited resources (computational power, time constraints, restriction of usable material) create additional difficulties to solving a given problem. Besides robot soccer, some other successful examples are: Robot juggling with a so-called devil-stick [6], box pushing with a special clustering technique [127], collecting and transporting small disks by a team of four robots in a decentralised way [133], and the huge area of roboters being part of a production line.

Chapter 6

Conclusion and Outlook

In the present work two-player zero-sum Markov games (2P-ZS-MGs) are shown to be an adequate framework for modeling robot soccer in contrast to the widely used Markov decision process (MDP) framework. Furthermore, a grid soccer model for an arbitrary resolution grid as well as for an arbitrary number of agents is provided. It seems to be well-suited for comparison of large scale 2P-ZS-MG effects caused by fine discretisation and multi-agent scenarios.

Amongst the theoretic aspects, the development of a notion of symmetry for 2P-ZS-MGs, particularly 2P-ZS-MG μ-homomorphisms, is to be accentuated. The concept is shown to be a non-trivial extension of recently developed MDP (1-) homomorphisms and its relation to the special case of classical group actions is studied. A qualitatively new class of symmetries which does not occur in MDPs and exchanges the two players of a 2P-ZS-MG is proven to fulfill some natural algebraic properties, especially that it can be composed with recently developed MDP symmetries. Practitioners already applied the results of this symmetry concept without a precise theoretical foundation. In the present thesis their work is legitimated mathematically a posteriori.

To also highlight some practical aspects the origination of the software package DRPOST is to be mentioned by which all numerical results are computed. Notably, it includes a new asynchronous dynamic progreamming (DP) algorithm called MaG-Clus-VI which intertwines global and local aspects of updating by means of almost invariant sets established in dynamical systems theory. The usefulness and usability of the software package aims at helping other researchers and practitioners to solve interesting problems and to gain deeper insights into 2P-ZS-MGs. Additionally, several comparative studies are performed by means of DRPOST: most notable are the aspects of incorporating dynamic programming (DP) methods – as typically not done by the reinforcement learning (RL) community – and discovering and investigating interesting phenomena such as the "max-min convergence boosting phenomenon" which is observed in 2P-ZS-MGs. Concerning realisability on physical robots – e. g. on AIBO ERS-7 – the author identifies the opponent localisation as the most urgent topic on which to focus future research.

However, solving problems raises others, and thus many ideas remain on how to continue research beyond this PhD thesis. In theory, many open questions exist if the models are differential games: How does discretisation affect the reliability of the solution? Does the limit of arbitrary fine discrete models converge to the continuous model? Most of the results in differential game theory are limited to the important but not general case of pursuit evasion games. Questions about how to design a *stochastic* differential game

theory are even more challenging since stochastic dynamical systems without any control are also an interesting topic of current research.

Not only continuous dynamical games but also generalising results from 2P-ZS-MGs to non-zero-sum and multi-player games with more than two players offer interesting open questions. A natural idea is to replace the solution of the matrix game by a more general Nash equilibrium solution of a non-zero-sum or multi-player game. However, this introduces quite a burden since all the issues about solution concepts for games with only a single state and single time step (e. g. selection of one Nash equilibrium if multiple exists) known to the game theory community become relevant for *every state and every time step* in the iterative procedure of finding a solution of a multiple time step multiple state game.

Practically challenging is the application of RL and DP methods to big problem instances. Some practical ideas which are partly biologically inspired and could help to manage such instances are according to Kaelbling [88]: *shaping*, i. e. first presenting simple problems and then raising the difficulty level little by little (e. g. reward shaping [169]), *imitation*, i. e. learning by watching other agents or providing parts of a policy with the aid of humans ([167, 204] utilise initial policies called experts to accelerate learning), and *reflexes*, i. e. providing some standard reactions to standard situations.[(1)]

What all these ideas have in common is to give up tabula rasa learning which is also the motivation of the present work. The difference is that in the three approaches above knowledge of a hand-coded (part of a) policy is provided: the experts are directly policies, reflexes can be interpreted as policies defined only on special states, and shaping can be performed by restricting the model and the policies to a subset of the original domain. To make progress in the spirit of giving up tabula rasa learning, however, the author suggests using DP methods with simpler models to provide an initial guess for a policy.[(2)] In future research, all these methods could be compared and intertwined e. g. by utilising the author's approach with different models of the same real world problem and let (nearly) optimal policies of each model be different experts to imitate.

Another suggestion of Kaelbling is to make *reinforcement signals local*. This addresses the model designing phase rather than the learning algorithms but may be advantageous especially in navigation tasks [133]. Parallels can be found to so-called potential respectively vector field approaches in which goals and special objects or locations produce a virtual vector field to influence the motion of a robot. Caution is adequate for vector field approaches as well as for local reward methods because unintendedly introduced subgoals may lead to suboptimal solutions and may prevent the agent from reaching the original goal. It is also interesting future work to design a suitable local reward function for the multi-player robot soccer model and to study the influence of such models on 2P-ZS-MGs.

Finally, a lot of work has been done for MDPs and in this area there are still open questions but most of the comparable work for 2P-ZS-MGs has not yet been completed. From a practical point of view, a great deal of function approximation methods succesfully applied to MDPs should also be evaluated in 2P-ZS-MGs. It is the author's hope that a multilaterally developed public software package could originate as a byproduct of future work. This should provide a full variety of MDP and 2P-ZS-MG standard example models, include an essential selection of function approximation and data mining methods, and

[(1)] Reflexes could speed up the beginning phase of learning in which nothing happens until a random walk detects some goal (or non-zero reward) and could help to avoid dangerous situations [138]. In robot soccer there are no dangerous situations but a walk close to a cliff or controlling a nuclear reactor provides a meaningful example.

[(2)] [69] shows that the transfer from one model to another (there: simulation to real robots) can work.

integrate symmetry reduction and hierarchical techniques in a modular way. By means of such a software tool, transparency and comparability amongst the huge variety of proposed learning methods and test models could be improved.

Appendix A

Basics of Group Homomorphisms and Group Actions

In this appendix some standard definitions of group homomorphisms and group actions are repeated. By Proposition A.4 and Proposition A.5 it is shown that equivalence relations and group actions are equivalent concepts. The proof of Proposition A.5 is given for the convenience of the reader. Some of the statements in this appendix also hold for infinite groups but only the application to finite groups is intended.

Before continuing with group actions a short reminder of group homomorphisms shall be given:

A.1 Definition (Group Homomorphism, Isomorphism [60])
Let $(G_1, *)$ and $(G_2, \odot)$ be two groups with group operation $*$ and $\odot$, respectively. A map $h : (G_1, *) \to (G_2, \odot)$ (or short $h : G_1 \to G_2$) is called a group homomorphism if

$$\forall g_1, \widetilde{g}_1 \in G_1 : \; h(g_1 * \widetilde{g}_1) = h(g_1) \odot h(\widetilde{g}_1)\,. \tag{A.1}$$

A (group) homomorphism is called isomorphism if it is a bijection.

A.2 Definition (Group Action, Transformation Group [24, 60])
Let G be a group and let X be a set. A (left) group action Θ is a map $\Theta : G \times X \to X$ with the properties:

$$\forall g, h \in G \; \forall x \in X : \; \Theta(g, \Theta(h, x)) = \Theta(gh, x) \tag{A.2}$$

and for the identity $1 \in G$ holds:

$$\forall x \in X : \; \Theta(1, x) = x\,. \tag{A.3}$$

A simplified notation for group actions will be $gh \cdot x = ghx = \Theta(g, \Theta(h, x))$ when there are no misunderstandings possible. The composition of group elements is due to the standard group operation. [60] states that for each $g \in G$ the map $\sigma_g : X \to X$, $a \mapsto \sigma_g(a) = g \cdot a$ is a *permutation* or *transformation* of X (i. e. bijection from X to X) and that the map from G to S_X defined by $g \mapsto \sigma_g$ is a homomorphism. This means that every element $g \in G$ acts on X as a permutation in a manner that is consistent with the group operation of G. To make this more precise define the *kernel of a group action* by $K = \{g \in G : \; (\forall x \in X : g \cdot x = x)\} \subseteq G$ and the action to be *faithful* if $K = \{e\}$. Then, the kernel K is a

normal subgroup of G and hence induces equivalence classes corresponding to elements of the quotient group G/K.

Furthermore, not only each group action induces a homomorphism by $g \mapsto \sigma_g$ but also reversely any homomorphism $\varphi : G \to S_X$ defines a group action of G on X by $g \cdot x = \varphi(g)(x)$ for all $g \in G$ and $x \in X$. The kernel of this group action is the kernel of φ and the permutation representation $g \mapsto \sigma_g$ equals φ. Thus, the following proposition holds:

A.3 Proposition (Characterising Group Actions [60])
For any group G and any non-empty set X there exists a bijection between the actions of G on X and the homomorphisms from G into S_X.

The following result finally relates group actions to equivalence classes:

A.4 Proposition (Equivalence Relations Induced by Group Actions [60])
For any group G acting on a non-empty set X the group action induces an equivalence relation on X by

$$x' \in [x] \quad \textit{iff} \quad \exists g \in G : \; x' = g \cdot x \,. \tag{A.4}$$

For each $x \in X$ the size of the equivalence class $|\,[x]\,| = |G : G_x|$ which is the index *of the* stabiliser $G_x = \{g \in G : g \cdot x = x\}$.

A shorter notation of the equivalence classes induced by a group action is $[x] = G \cdot x = \{g \cdot x : g \in G\}$. $G \cdot x$ is also called the *group orbit* of x under G.

Proposition A.4 shows that each group action induces an equivalence relation on X. Moreover, equivalence relations on X also induce group actions stated by Proposition A.5. If equivalence classes of all group actions which induce the same equivalence classes on X are formed, then group actions and equivalence relations on X are equivalent concepts.

A.5 Proposition (Group Actions Induced by Equivalence Relations [68])
Let X be a non-empty set with an equivalence relation, i. e. a partition of equivalence classes $\mathcal{P}(X) = \{[x] : x \in X\}$, and let G be the set of all bijective functions $g : X \to X$ that preserve the structure of the equivalence classes, i. e.

$$\forall g \in G \; \forall x \in X : \; g(x) \in [x] \,. \tag{A.5}$$

Then $(G, \circ)$ is a group with $\circ$ being the standard composition of functions and the group action $\Theta : G \times X \to X$, $\Theta(g, x) = g(x)$ induces equivalence classes equal to that of $\mathcal{P}(X)$.

Proof: The proof of [68] is adapted to the notion of group actions. G is a group because it contains the identity, and the inverses and compositions of $g_i \in G$ are also members of G because they all operate with respect to the partition $\mathcal{P}(X)$. Furthermore, Θ is a group action.
It remains to show: $\forall x \in X : \; G \cdot x = [x]$. The inclusion $G \cdot x \subseteq [x]$ directly follows from Equation A.5. To prove $G \cdot x \supseteq [x]$, recall that $x \in G \cdot x$ and observe that for any $y \in [x]$ the swapping function $g_{x,y} : X \to X$, defined by $g_{x,y}(x) = y$, $g_{x,y}(y) = x$, and otherwise $g_{x,y}(z) = z$, is an element of G. Thus, $y \in G \cdot x$ which completes the proof. □

Appendix B

Bellman Equations and Iterative Linear Solvers

In this appendix connections between standard iterative linear solvers [214] such as Jacobi or Gauss-Seidel method are related to the linear versions of the Bellman equation which emerges if the policies are fixed. In principle, this does not reduce the problem of determining the optimal value function but gives a possibility for an a posteriori analysis of the iterative scheme. The reason is that after a non-linear Bellman update the optimal policies are known and the non-linear update can be reformulated as a linear one with the calculated optimal policies. The idea is the same as for MDPs [168].

Iterative Solution of Linear Equations. The problem of solving a linear equation of the type

$$Ax = b, \quad x, b \in \mathbb{R}^n, \ A \in \mathbb{R}^{n,n} \tag{B.1}$$

can be solved iteratively which is especially useful for large systems of equations. The standard idea is to rewrite the above equation to the fixed point formulation

$$x = C^{-1}b - C^{-1}(A - C)x \tag{B.2}$$

for some invertible matrix $C \in \mathbb{R}^{n,n}$. The corresponding iterative method is simply

$$x_{m+1} = C^{-1}b - C^{-1}(A - C)x_m \tag{B.3}$$

with some initial x_0 and convergence of the method depends on the spectral properties of the *update matrix* $C^{-1}(A - C)$ (spectral radius less than 1). For the classical Jacobi method is $C = D$ where A is written as a sum of its diagonal, lower triangle, and upper triangle part:

$$A = D + L + U. \tag{B.4}$$

The standard Gauss-Seidel method utilises $C = D + L$.

Bellman Equation as Linear Equation. As mentioned above the idea is the same as for MDPs with the difference that in 2P-ZS-MGs the policies of both players have to be fixed to make the Bellman equation linear. Equations 2.48 and 2.49 with the minimum equivalently also taken over probability distributions become for fixed policies π_1, π_2 which fulfill the max-min property:

$$V_{k+1}(s) = \sum_{a \in \mathcal{A}(s)} \sum_{o \in \mathcal{O}(s)} \pi_{1,s}(a) \cdot \pi_{2,s}(o) \cdot Q_k(s, a, o) \tag{B.5}$$

where

$$Q_k(s,a,o) = R(s,a,o) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,o,s') \cdot V_k(s'). \tag{B.6}$$

All in all, this is equivalent to performing a single iterative step of solving a system of linear equations $Ax = b$ for each value iteration step. At iterate $k+1$ the variable reads to $x_s = V_{k+1}(s)$ with the initial guess estimate being V_k, the right-hand side is defined by the *modified reward*

$$b_s = \sum_{a \in \mathcal{A}(s)} \sum_{o \in \mathcal{O}(s)} \sum_{s' \in \mathcal{S}} \pi_{1,s}(a) \cdot \pi_{2,s}(o) \cdot T(s,a,o,s') \cdot R(s,a,o,s') \tag{B.7}$$

for rewards possibly depending on the future states s', and the matrix $A = I - \gamma T_{s,s'}$ depends on the *transition matrix*

$$T_{s,s'} = \sum_{a \in \mathcal{A}(s)} \sum_{o \in \mathcal{O}(s)} \pi_{1,s}(a) \cdot \pi_{2,s}(o) \cdot T(s,a,o,s')\,. \tag{B.8}$$

With these ingredients the normal DP update is specified in terms of iterative solvers for linear systems by $C = I$ and the Gauss-Seidel type update by $C = I + L$. Thus, the methods are typically different from the versions for iterative linear solvers unless $D = I$, i. e. that the transition matrix $T_{s,s'}$ has zero probability transitions from each state to itself.

Finally, note again that the policies π_1, π_2 are not known a priori but determined during value iteration such that an a posteriori analysis is possible. However, the convergence speed of DP methods is independent from the complexity of determining π_1, π_2 but rather dependent on the convergence properties of the matrix A. Furthermore, for every iteration a different matrix and right hand side is to be calculated such that the convergence results (contractivity of the update matrix) can differ from step to step.

Appendix C

The Software Package DRPOST

C.1 Introduction

The software package DRPOST (Discrete Robust Probabilistic Optimal Strategy Tool) developed during the PhD thesis is also used for computing the results of Section 5.2. In this appendix, the basic structure of the package is described. First, the files with ending `.m` are MATLAB files (MATLAB 7.3.0.298 was used but the basic files should also work on previous MATLAB versions). When starting MATLAB for the first time all subdirectories can be added to the MATLAB path while the first call of `renew_model.m` will clear archive subdirectories and unused model directories from the path. The directory `phd_scripts` contains executable scripts which generate material contained in this thesis. They can be considered a good starting point to learn how the software package works and which parameters are important.

For the sake of completeness, the other subdirectories are shortly described:

`data_matlab` contains MATLAB save files (ending `.mat`) for the most important or lengthy computations by scripts in the directory `phd_scripts`.

`func_approx` contains the structure for inserting arbitrary function approximation schemes. The approximation procedures can also be externally provided e. g. by the MATLAB neural net toolbox.

`func_approx_EXTERN` is empty but the intended directory for external software packages (not written by the author) which are specialised to function approximation.

`models` contain one subdirectory for each model. The name of the currently used model can be specified by the MATLAB struct field `DP_RL_param.model_dir`. This model will be copied to `model_current` by the function `renew_model.m` and the MATLAB path is adapted such that only this model belongs to it.

`phd_scripts` contains all scripts for computations in this thesis as mentioned above.

`pictures` is used for storage of pictures and figures.

`tools_divers` is a collection of different functions not fitting to one of the other categories. It contains a subdirectory `policy` with policy generating and modifying functions and a subdirectory `plot` for general plot routines.

The use in MATLAB is quite straightforward because every function and script is documented and a complete help on how to call this function is displayed in MATLAB as usual by `help <functionname>`. Furthermore, only the structs `DP_RL_param`, `DP_RL_prev_step`,

`param_model`, `param_model_large`, `simulator_param`, and `strat_all` are needed to keep all information, and all these structs as well as value functions and strategies (function approximation structs) have a subfield `.info` which gives all necessary information about the structs.

C.2 Technical Aspects of Symmetry Reduction in 2P-ZS-MGs

Efficient Data Structures for State Spaces. In general, all available states $s \in \mathcal{S}$ have to be made accessible by computer software. The most obvious and easiest way in terms of programming effort and clarity is to store a list of all characteristics for every state, i. e. for grid soccer all elements $\subseteq \mathbb{N}^{2(n_a+n_o+1)}$ which describe a discrete soccer state. However, if the state space $\mathcal{S}$ is very large as e. g. in a multi-player grid soccer with many robots it may be important to represent the list of states in a compact way. In the following, *hash functions* are not considered to be a sensible solution because they are not injective.

A very compact way for finite 2P-ZS-MGs is to assign a *list of states* (state-actions) which is just a mapping $i_{\text{no}} : \mathcal{S} \to \mathbb{N}$ ($i_{\text{no}} : \mathcal{SAO} \to \mathbb{N}$) preferably such that the smallest possible integer numbers, i. e. all numbers from 1 to $|\mathcal{S}|$ (1 to $|\mathcal{SAO}|$), are assigned. This number representation introduces the cost of computing the function i_{no} *very* often and, hence, should be very simple. The reason for the immense use is that the assignment of value functions and policies, the evaluation of the transition function which includes the determination of all possible following states, and the evaluation of the reward function need the evaluation of i_{no} for every state or state-action.[(1)] Sometimes, the inverse i_{no}^{-1} also has to be computed e. g. for interpreting a value function or policy stored in the number representation.

Efficient Data Structures for Multi-Player Grid Soccer. The structure of possible states in robot soccer which is simply the product of discrete intervals[(2)] makes it possible to find a reasonable number representation of $\mathcal{S}$ by simple `for`-loops and multiplications while the calculation of the inverse i_{no}^{-1} needs divisions with remainder. Based on the enumeration of states, $\mathcal{SAO}$ can be enumerated by storing (a vector of) the number of possible actions in every state.[(3)]

Now, if symmetries are introduced the aim is to store all symmetric states by only one entry to reduce the amount of stored data. The main problem is that the function i_{no} typically becomes quite complicated and no simple computation comparable to the case without symmetries is obvious. Furthermore, the standard way to store the mapping directly in a lookup list is impractical by reason of size. Thus, a new way of storing value functions is suggested by the author: employing sparse matrices. *Sparse matrices* are a well-known structure that are typically utilised to store large matrices with many zero elements. Sparsity is often defined by the fact that the number of non-zero elements grow linearly with matrix size instead of quadratically which would be natural. Sparsity does not matter to the present work, it should only be mentioned that in principle in a sparse matrix only non-zero entries are stored together with their row and column number. To

[(1)] For a small model it could be more practical to store the transition function and reward function once for the number representation and than work on that data. However, if the state space is large (as always assumed) then the transition function will typically be much larger.

[(2)] A *discrete interval* is an intersection of a real interval I with $\mathbb{N}_0$.

[(3)] In grid soccer, actions for robots at the margin of the soccer field and the number of kicks can vary. If the action space would not vary $\mathcal{SAO}$ could be handled in exactly the same way as $\mathcal{S}$.

read a value the index lists are searched for matching entries and, if no match is found, a zero is returned.

In the context of symmetries this means that the enumeration of the complete state(-action) space can be initialised for storing a value function or a policy, then each state(-action) is mapped to a unique representative of its equivalence class (see Section 3.2), and finally the value function or policy *only* of the representatives are accessed. The main advantage of this approach is twofold: Firstly, there is no need to compute i_{no} directly, it is simply stored by the matrix entry list. Secondly, by the size of the matrix (more columns or more rows) a trade-off between access speed and used memory of the sparse matrix can be decided. The reason is that the size of the matrix influences the length of vectors of the row and column information which typically is stored as a vector of vectors. If the storage is first row and then column and a matrix is already a row or a column vector then the maximum number of pointers to columns has to be stored (storage bad, access fast) or only one (storage good, access slow). The best compromise seems to be a quadratic matrix. A generalisation to a multi-level tree structure with a variable number of leaves at each level which could even make i_{no} superfluous is not considered because for the multi-player grid soccer example a sparse vector is already sufficiently effective.

Appendix D

Detailed Tables of Numerical Results

The numerical results in this section are obtained by means of the software package DRPOST by scripts gathered in the subdirectory `phd_scripts`. The material is omitted in the main part because not all results are spectacular. However, the author regards it as his duty to provide this material and hopes that it may be helpful.

D.1 Initial Value Functions V_0 and Discount Factors γ

The following tables are related to Section 5.2.5[(1)] and give detailed information about all combinations of initial value function V_0, discount factor γ, Gauss-Seidel and symmetry reduction types, and max-min, max, or fixed policy DP methods.

For reasons of comparability and clear view the tables are placed in combinations of three tables per page with the following ordering principle: the following page contains three tables of max-min DP methods, the subsequent page three tables of max DP methods and the next page three tables of fixed strategy DP methods (policy evaluation), whereas on each page the first table contains data about a non Gauss-Seidel method without symmetry reduction, the second table about a Gauss-Seidel method without symmetry reduction, and the third table about a Gauss-Seidel method with symmetry reduction. The fourth case of a non Gauss-Seidel method with symmetry reduction would yield the same results as the method on a non symmetry reduced model but simply updating each state in an equivalence class separately.

[(1)] A more detailed description of the setting and the interpretation of the basic facts can be found there.

	$V_0 \equiv 3$	$V_0 \equiv 1$	$V_0 \equiv 0.1$	$V_0 \equiv 0$	$V_0 \equiv -0.1$	$V_0 \equiv -1$	$V_0 \equiv -3$
$\gamma = 0.10$	4 0.00083	4 0.00043	4 0.00025	4 0.00023	4 0.00024	4 0.00043	4 0.00083
$\gamma = 0.25$	7 0.00053	7 0.00029	6 0.00070	6 0.00066	6 0.00070	7 0.00029	7 0.00053
$\gamma = 0.50$	13 0.00075	12 0.00050	11 0.00038	11 0.00028	11 0.00038	12 0.00050	13 0.00075
$\gamma = 0.75$	31 0.00087	27 0.00091	19 0.00091	13 0.00015	19 0.00090	27 0.00091	31 0.00086
$\gamma = 0.90$	88 0.00096	77 0.00099	56 0.00094	30 0.00038	55 0.00100	77 0.00099	88 0.00096

Table D.1: Comparison of different initial value functions V_0 and discount factors γ for a grid soccer model *without symmetry reduction* and a *max-min* value iteration method *with standard updates* (not Gauss-Seidel). In each cell of the table the number of needed iteration steps and the yielded precision ε with a stopping criterion precision of $\varepsilon = 1 \cdot 10^{-3}$ (ε as in Corollary 4.3) and a matrix game solution precision of $1 \cdot 10^{-6}$.

	$V_0 \equiv 3$	$V_0 \equiv 1$	$V_0 \equiv 0.1$	$V_0 \equiv 0$	$V_0 \equiv -0.1$	$V_0 \equiv -1$	$V_0 \equiv -3$
$\gamma = 0.10$	4 0.00022	4 0.00022	4 0.00022	4 0.00023	4 0.00025	4 0.00043	4 0.00085
$\gamma = 0.25$	6 0.00054	6 0.00062	6 0.00065	6 0.00066	6 0.00071	7 0.00025	7 0.00057
$\gamma = 0.50$	11 0.00050	10 0.00063	10 0.00051	10 0.00050	10 0.00039	12 0.00055	13 0.00049
$\gamma = 0.75$	24 0.00083	22 0.00089	20 0.00072	20 0.00057	21 0.00042	24 0.00090	27 0.00098
$\gamma = 0.90$	72 0.00095	66 0.00096	49 0.00095	43 0.00095	51 0.00098	70 0.00096	77 0.00099

Table D.2: Comparison of different initial value functions V_0 and discount factors γ for a grid soccer model *without symmetry reduction* and a *max-min* value iteration method *with Gauss-Seidel updates* (standard enumeration of states). Description of entries as in Table D.1.

	$V_0 \equiv 3$	$V_0 \equiv 1$	$V_0 \equiv 0.1$	$V_0 \equiv 0$	$V_0 \equiv -0.1$	$V_0 \equiv -1$	$V_0 \equiv -3$
$\gamma = 0.10$	4 0.00011	4 0.00011	4 0.00020	4 0.00023	4 0.00021	4 0.00022	4 0.00067
$\gamma = 0.25$	5 0.00058	5 0.00034	5 0.00059	5 0.00066	5 0.00060	6 0.00025	6 0.00090
$\gamma = 0.50$	9 0.00055	9 0.00042	8 0.00027	8 0.00001	8 0.00016	9 0.00055	10 0.00058
$\gamma = 0.75$	14 0.00094	14 0.00089	13 0.00059	12 0.00006	15 0.00093	21 0.00093	24 0.00072
$\gamma = 0.90$	30 0.00087	29 0.00086	28 0.00087	26 0.00037	49 0.00100	66 0.00097	71 0.00099

Table D.3: Comparison of different initial value functions V_0 and discount factors γ for a grid soccer model *with symmetry reduction* and a *max-min* value iteration method *with Gauss-Seidel updates* (standard enumeration of states). Description of entries as in Table D.1.

	$V_0 \equiv 3$	$V_0 \equiv 1$	$V_0 \equiv 0.1$	$V_0 \equiv 0$	$V_0 \equiv -0.1$	$V_0 \equiv -1$	$V_0 \equiv -3$
$\gamma = 0.10$	4 0.00062	4 0.00022	4 0.00020	4 0.00022	4 0.00024	4 0.00042	4 0.00082
$\gamma = 0.25$	7 0.00037	6 0.00052	6 0.00061	6 0.00066	6 0.00070	7 0.00029	7 0.00053
$\gamma = 0.50$	13 0.00074	11 0.00099	11 0.00100	12 0.00055	12 0.00060	13 0.00053	14 0.00051
$\gamma = 0.75$	31 0.00080	26 0.00099	26 0.00090	27 0.00077	27 0.00086	29 0.00083	32 0.00079
$\gamma = 0.90$	89 0.00096	60 0.00096	80 0.00099	81 0.00099	82 0.00099	88 0.00098	95 0.00097

Table D.4: Comparison of different initial value functions V_0 and discount factors γ for a grid soccer model *without symmetry reduction* and a *max* value iteration method *with standard updates* (not Gauss-Seidel). In each cell of the table the number of needed iteration steps and the yielded precision ε with a stopping criterion precision of $\varepsilon = 1 \cdot 10^{-3}$ (ε as in Corollary 4.3) and a matrix game solution precision of $1 \cdot 10^{-6}$.

	$V_0 \equiv 3$	$V_0 \equiv 1$	$V_0 \equiv 0.1$	$V_0 \equiv 0$	$V_0 \equiv -0.1$	$V_0 \equiv -1$	$V_0 \equiv -3$
$\gamma = 0.10$	4 0.00053	4 0.00018	4 0.00005	4 0.00006	4 0.00006	4 0.00013	4 0.00025
$\gamma = 0.25$	6 0.00088	6 0.00030	5 0.00021	5 0.00025	5 0.00029	5 0.00064	6 0.00027
$\gamma = 0.50$	11 0.00041	9 0.00100	7 0.00089	7 0.00079	8 0.00037	9 0.00044	10 0.00039
$\gamma = 0.75$	21 0.00066	18 0.00070	14 0.00078	15 0.00081	16 0.00069	18 0.00085	20 0.00078
$\gamma = 0.90$	53 0.00099	40 0.00099	47 0.00095	48 0.00093	48 0.00097	52 0.00093	55 0.00099

Table D.5: Comparison of different initial value functions V_0 and discount factors γ for a grid soccer model *without symmetry reduction* and a *max* value iteration method *with Gauss-Seidel updates* (standard enumeration of states). Description of entries as in Table D.4.

	$V_0 \equiv 3$	$V_0 \equiv 1$	$V_0 \equiv 0.1$	$V_0 \equiv 0$	$V_0 \equiv -0.1$	$V_0 \equiv -1$	$V_0 \equiv -3$
$\gamma = 0.10$	4 0.00053	4 0.00018	4 0.00005	4 0.00005	4 0.00006	4 0.00012	4 0.00037
$\gamma = 0.25$	6 0.00086	6 0.00023	5 0.00020	5 0.00025	5 0.00029	5 0.00081	6 0.00032
$\gamma = 0.50$	10 0.00097	9 0.00079	7 0.00083	7 0.00059	7 0.00088	9 0.00039	10 0.00035
$\gamma = 0.75$	20 0.00090	18 0.00063	14 0.00060	15 0.00066	15 0.00092	18 0.00068	19 0.00098
$\gamma = 0.90$	52 0.00094	40 0.00094	45 0.00095	46 0.00093	46 0.00097	49 0.00099	53 0.00096

Table D.6: Comparison of different initial value functions V_0 and discount factors γ for a grid soccer model *with symmetry reduction* and a *max* value iteration method *with Gauss-Seidel updates* (standard enumeration of states). Description of entries as in Table D.4.

	$V_0 \equiv 3$	$V_0 \equiv 1$	$V_0 \equiv 0.1$	$V_0 \equiv 0$	$V_0 \equiv -0.1$	$V_0 \equiv -1$	$V_0 \equiv -3$
$\gamma = 0.10$	4 0.00062	4 0.00022	3 0.00050	3 0.00030	3 0.00050	4 0.00022	4 0.00062
$\gamma = 0.25$	7 0.00038	6 0.00053	5 0.00038	4 0.00085	5 0.00038	6 0.00053	7 0.00038
$\gamma = 0.50$	13 0.00075	12 0.00052	9 0.00063	8 0.00055	9 0.00063	12 0.00052	13 0.00075
$\gamma = 0.75$	31 0.00087	27 0.00092	20 0.00090	17 0.00071	20 0.00090	27 0.00092	31 0.00087
$\gamma = 0.90$	88 0.00095	77 0.00099	57 0.00097	44 0.00100	57 0.00097	77 0.00099	88 0.00095

Table D.7: Comparison of different initial value functions V_0 and discount factors γ for a grid soccer model *without symmetry reduction* and a *fixed* value iteration method *with standard updates* (not Gauss-Seidel). In each cell of the table the number of needed iteration steps and the yielded precision ε with a stopping criterion precision of $\varepsilon = 1 \cdot 10^{-3}$ (ε as in Corollary 4.3) and a matrix game solution precision of $1 \cdot 10^{-6}$.

	$V_0 \equiv 3$	$V_0 \equiv 1$	$V_0 \equiv 0.1$	$V_0 \equiv 0$	$V_0 \equiv -0.1$	$V_0 \equiv -1$	$V_0 \equiv -3$
$\gamma = 0.10$	4 0.00033	4 0.00011	3 0.00041	3 0.00031	3 0.00032	4 0.00011	4 0.00033
$\gamma = 0.25$	6 0.00044	5 0.00085	5 0.00016	4 0.00089	4 0.00089	5 0.00092	6 0.00045
$\gamma = 0.50$	10 0.00090	9 0.00083	7 0.00097	7 0.00062	7 0.00091	9 0.00086	10 0.00091
$\gamma = 0.75$	21 0.00093	19 0.00085	15 0.00082	13 0.00090	14 0.00091	19 0.00080	21 0.00091
$\gamma = 0.90$	55 0.00093	49 0.00096	38 0.00096	33 0.00091	35 0.00094	49 0.00092	55 0.00092

Table D.8: Comparison of different initial value functions V_0 and discount factors γ for a grid soccer model *without symmetry reduction* and a *fixed* value iteration method *with Gauss-Seidel updates* (standard enumeration of states). Description of entries as in Table D.7.

	$V_0 \equiv 3$	$V_0 \equiv 1$	$V_0 \equiv 0.1$	$V_0 \equiv 0$	$V_0 \equiv -0.1$	$V_0 \equiv -1$	$V_0 \equiv -3$
$\gamma = 0.10$	4 0.00038	4 0.00012	3 0.00041	3 0.00031	3 0.00032	4 0.00013	4 0.00039
$\gamma = 0.25$	6 0.00046	5 0.00095	5 0.00016	4 0.00089	4 0.00090	6 0.00016	6 0.00047
$\gamma = 0.50$	10 0.00079	9 0.00076	7 0.00090	7 0.00060	7 0.00089	9 0.00079	10 0.00079
$\gamma = 0.75$	21 0.00081	19 0.00075	15 0.00074	13 0.00083	14 0.00085	19 0.00071	21 0.00079
$\gamma = 0.90$	54 0.00099	49 0.00091	38 0.00093	33 0.00091	35 0.00092	48 0.00098	54 0.00098

Table D.9: Comparison of different initial value functions V_0 and discount factors γ for a grid soccer model *with symmetry reduction* and a *fixed* value iteration method *with Gauss-Seidel updates* (standard enumeration of states). Description of entries as in Table D.7.

D.2 Additional Figures and Tables for the Comparative Studies of DP and RL methods

Initial Value Functions V_0 and Discount Factors γ

Figure D.1 shows the omitted results for fixed strategy DP methods related to Figure 5.7. The figure was omitted since it yields only expected results analogue to the max methods.

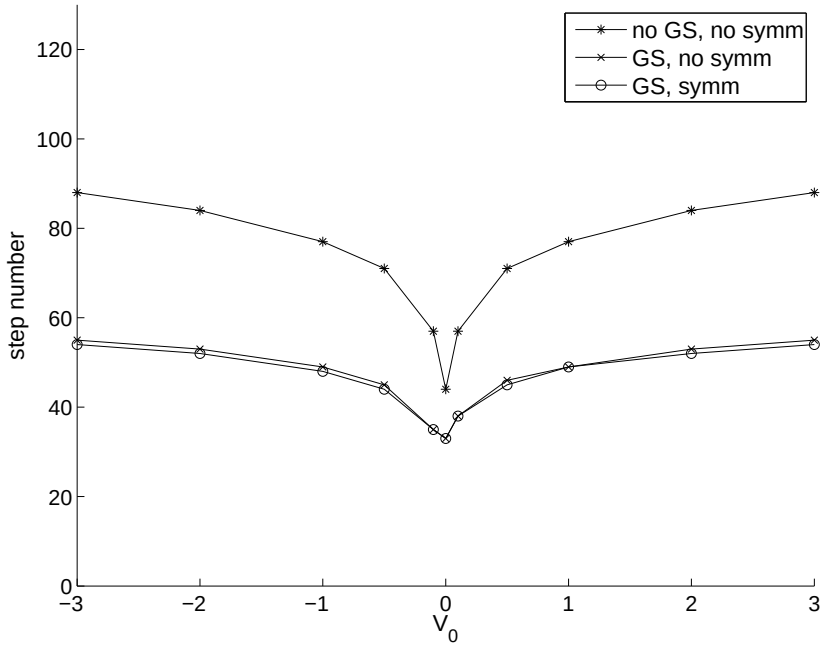


Figure D.1: Comparison of different Gauss-Seidel types with or without symmetry reduction: number of iteration steps over initial values of the initial value function V_0 for a 1v1 multi-player grid soccer model and a *fixed* strategy method without sorting strategy ($\gamma = 0.9$, stopping criterion precision $\varepsilon = 1 \cdot 10^{-3}$ (Corollary 4.3)).

Figure D.2 shows the omitted results for fixed strategy DP methods related to Figure 5.8. The figure was omitted since it yields only expected results analogue to the max-min and max methods.

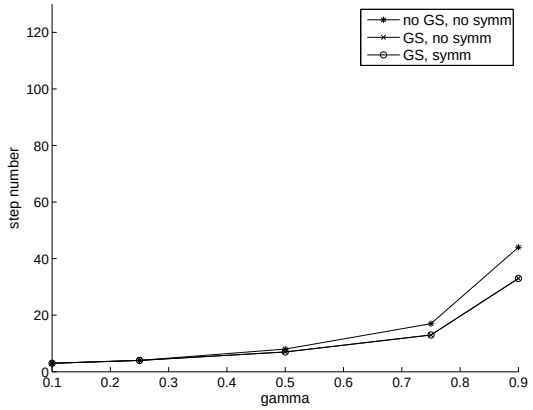


Figure D.2: Comparison of different Gauss-Seidel types with or without symmetry reduction: number of iteration steps over the discount factor γ for a 1v1 multi-player grid soccer model and a *fixed* strategy method without sorting strategy ($V_0 = 0$, stopping criterion precision $\varepsilon = 1 \cdot 10^{-3}$ (Corollary 4.3)).

Figure D.3 shows the omitted results for fixed strategy DP methods related to Figure 5.9. The figure was omitted since it yields only expected results analogue to the max methods.

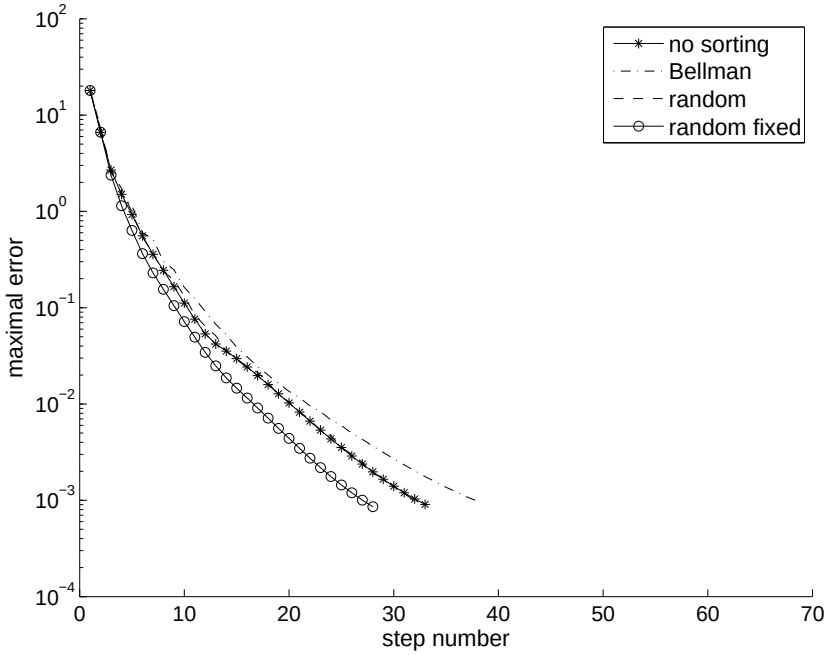


Figure D.3: Fixed policy method, Gauss-Seidel type, with symmetry reduction: Maximal DP error (logarithmic scale) over the number of iteration steps for a 1v1 multi-player grid soccer model ($V_0 = 0$, $\gamma = 0.9$, stopping criterion precision $\varepsilon = 1 \cdot 10^{-3}$ (Corollary 4.3)) by means of standard updates (no sorting), Bellman error estimation (Bellman), randomly rearranged state order in each iteration (random), and keeping the randomly rearranged state order of the first iteration (random fixed).

Strategy Evaluation Tables for 12×8 Grid Soccer

Additional Tables of evaluating different soccer strategies according to that of Section 5.2 are presented for the 1v1 grid soccer on a 12×8 field. These tables reveal only expected results and are not intended – like some other tables – to stress differences of a coarse to a medium discretisation of the soccer field. However, the time it would take to compute these results again by the software package DRPOST can be saved by looking onto this appendix.

	$\pi_3 = \mathrm{M}_{\mathrm{QL}}(\pi_1)$	$\pi_4 = \mathrm{M}_{\mathrm{VI}}(\pi_1)$	$\pi_5 = \mathrm{M}_{\mathrm{QL}}(\pi_2)$	$\pi_6 = \mathrm{M}_{\mathrm{VI}}(\pi_2)$
$\pi_1 = \mathrm{M}_{\mathrm{QL}}(\mathrm{M}_{\mathrm{QL}}(\mathrm{R}))$	$V:$ 0.225 $g_t:$ 0.021 $g_1:$ 0.991	$V:$ 0.234 $g_t:$ 0.022 $g_1:$ 0.985		
$\pi_2 = \mathrm{M}_{\mathrm{QL}}(\mathrm{M}_{\mathrm{VI}}(\mathrm{R}))$			$V:$ 0.233 $g_t:$ 0.022 $g_1:$ 0.980	$V:$ 0.236 $g_t:$ 0.022 $g_1:$ 0.984

Table D.10: Robustness of max-policies against worst-case opponents (12×8 soccer field) with better initial training partners. The explanation of how to read the table is as in Table 5.3.

	$\pi_2 = \mathrm{M}_{\mathrm{VI}}(\pi_1)$	$\pi_3 = \mathrm{MM}_{\mathrm{VI}}(R)$
$\pi_1 = \mathrm{MM}_{\mathrm{VI}}(\mathrm{R})$	$V:$ 0.000 $g_t:$ 0.023 $g_1:$ 0.505	$V:$ 0.000 $g_t:$ 0.023 $g_1:$ 0.503

Table D.11: Exploitability of optimal max-min opponents (12×8 soccer field). The explanation of how to read the table is as in Table 5.3.

	$\pi_1 = \mathrm{R}$	equal
$\pi_1 = \mathrm{R}$	$V:\ 0.000$ $g_t:\ 0.000$ $g_1:\ 0.577$	$V:\ 0.000$ $g_t:\ 0.000$ $g_1:\ 0.516$
$\pi_2 = \mathrm{M}_{\mathrm{QL}}(\mathrm{R})$	$V:\ -0.252$ $g_t:\ 0.023$ $g_1:\ 0.001$	$V:\ 0.000$ $g_t:\ 0.034$ $g_1:\ 0.500$
$\pi_3 = \mathrm{M}_{\mathrm{VI}}(\mathrm{R})$	$V:\ -0.252$ $g_t:\ 0.023$ $g_1:\ 0.000$	$V:\ 0.000$ $g_t:\ 0.021$ $g_1:\ 0.501$
$\pi_4 = \mathrm{MM}_{\mathrm{QL}}(\mathrm{R})$	$V:\ -0.179$ $g_t:\ 0.016$ $g_1:\ 0.001$	$V:\ -0.000$ $g_t:\ 0.017$ $g_1:\ 0.505$
$\pi_5 = \mathrm{MM}_{\mathrm{VI}}(\mathrm{R})$	$V:\ -0.173$ $g_t:\ 0.016$ $g_1:\ 0.001$	$V:\ 0.000$ $g_t:\ 0.023$ $g_1:\ 0.507$
$\pi_6 = \mathrm{M}_{\mathrm{QL}}(\pi_2)$	$V:\ -0.002$ $g_t:\ 0.000$ $g_1:\ 0.110$	$V:\ 0.000$ $g_t:\ 0.001$ $g_1:\ 0.468$
$\pi_7 = \mathrm{M}_{\mathrm{QL}}(\pi_3)$	$V:\ -0.002$ $g_t:\ 0.000$ $g_1:\ 0.132$	$V:\ 0.000$ $g_t:\ 0.000$ $g_1:\ 0.485$

Table D.12: Analysis of offensiveness and defensiveness of different policies (12×8 soccer field). The explanation of how to read the table is as in Table 5.3.

List of Figures

List of Tables

Glossary

$\emptyset$	empty set
$\overline{A}$	complement of a set A
$A \cap B$	intersection of two sets A and B
$A \cup B$	union of two sets A and B
$A \setminus B$	difference of two sets: set A minus set B
$A \times B$	product of two sets A and B
A^T	transpose of a matrix or vector A
2P-ZS-MG	Two Player Zero Sum Markov Game $\mathcal{M}$
$\mathcal{A}(s)$	set of actions in a state s for a Markov decision process or for the first agent P_1 of a two-player zero-sum Markov game
AI	Artificial Intelligence
α	possibly time- and state-dependent learning rate of an reinforcement learning algorithm
$\mathrm{Aut}(\mathcal{M})$	group of automorphisms (also: symmetry group) of a set or especially of a Markov decision process or a two-player zero-sum Markov game
B	box (generalised rectangle) used for a special class of set oriented numerical methods
$\mathcal{B}_{\mathrm{MDP}}$	Bellman operator for a Markov decision process
$\mathcal{B}_{\mathrm{MG}}$	Bellman operator for a two-player zero-sum Markov game
$\widetilde{\mathcal{B}}_{\mathrm{MG}}$	numerical approximation of a Bellman operator for a two-player zero-sum Markov game
$BR_i(\pi_{-i})$	best response or best reply policy of agent P_i to the joint policy π_{-i} of all other agents
C_{ext}	external costs of a subset of vertices of a graph
χ_A	characteristic function on a set A with $\chi_A(x)$ being 1 if $x \in A$ and 0 else
C_{int}	internal costs of a subset of vertices of a graph
C^n	n-times continuously differentiable functions
$\mathcal{D}$	decision epoch for a Markov decision process or a two-player zero-sum Markov game
∂_{ij}	Kronecker symbol (1 if i equals j else 0)
∂S	boundary of a set S
d_Γ	game matrix distance
DP	Dynamic Programming

$\frac{\mathrm{d}x}{\mathrm{d}t}$	total derivative of x with respect to t
E	edge set of a graph G
e	Euler's number G
$\mathrm{E}\{X\}$	expectation of random variable X
$\mathrm{E}_\pi\{X\}$	expectation of a random variable X in a Markov decision process or a two-player zero-sum Markov game if the (joint) policy π is executed
e_i	i-th unit vector of $\mathbb{R}^n$
$\mathcal{E}_V(i+1)$	error bound for numerical approximation of value iteration in 2P-ZS-MGs
$\widetilde{f}$	approximation to a function f
G	a graph with vertex set V and edge set E
Γ	a game
γ	discount factor for long-term rewards in a Markov decision process or a two-player zero-sum Markov game
id_X	identity map on a set X
i_{no}	mapping of state or state-action space to a number representation
LP	Linear Programming
LSPI	Least Squares Policy Iteration
$[M]$	matrix game with game matrix M
$\mathcal{M}$	space of (signed) measures
MAS	Multi Agent System
$\mathcal{M}_{\mathcal{B}}$	space of (signed) measures discretised by a collection of boxes $\mathcal{B}$
MDP	Markov Decision Process $\mathcal{M}$ or sometimes also Markov Decision Problem
MGR	Matrix Game Reduction
μ	a measure (sometimes especially the Lebesgue one)
μ_{Leb}	Lebesgue measure (sometimes abbreviated by μ)
$\mathbb{N}$	set of natural numbers
$\mathbb{N}_0$	set of natural numbers including zero
n_a	number of robots in the so-called first soccer team
n_o	number of robots in the so-called second or opponent soccer team
NP	non-deterministic polynomial: measure of hardness of an algorithmic problem are e. g. NP-hard or NP-complete
$\mathcal{O}(s)$	set of actions in a state s for the second agent P_2 of a two-player zero-sum Markov game
$\mathcal{O}_\varphi(x)$	orbit or trajectory of a dynamical system with respect to a flow φ
P	transfer operator of a dynamical system
$P_{\mathcal{B}}$	transfer operator of a dynamical system discretised with respect to a partition (of boxes) $\mathcal{B}$

$\mathcal{P}(A)$	partition of a set A into disjoint subsets (measure zero of pairwise intersections)
$\mathrm{PD}(B)$	set of all probability distributions on a (Borel) set B
φ	flow of a dynamical system
P_i	player of a matrix game (also called agent)
Π_x	projection of a tuple onto the x component
$\Pi^*(\mathrm{V})$	optimal policy of a Markov decision process or a two-player zero-sum Markov game with respect to state value function V
π	policy of a Markov decision process or a two-player zero-sum Markov game (a subindex i indicates the agent P_i)
π_s	policy of a Markov decision process or a two-player zero-sum Markov game restricted to a state s
π^*	optimal policy of a Markov decision process or a two-player zero-sum Markov game (a subindex i indicates the agent P_i)
π_{-i}	joint policy of all agents with exception of agent P_i
PO-MDP	Partially Observable Markov Decision Process (see also MDP)
$\mathrm{Prob}\{X\}$	probability that event X happens
$\mathrm{Prob}\{X \mid Y\}$	conditional probability that event X happens under the condition Y
$\mathbb{Q}$	set of rational numbers
Q^π	state action (or Q-) value function of policy π for a Markov decision process or a two-player zero-sum Markov game
$Q^*(= Q^{\pi^*})$	optimal value function for a Markov decision process or a two-player zero-sum Markov game
$\mathbb{R}$	set of real numbers
R	reward function for a Markov decision process or a two-player zero-sum Markov game
R	reward or payoff matrix for a matrix game (two matrices R_1 and R_2 in a bimatrix game)
$\mathcal{R}$	return of a Markov decision process or a two-player zero-sum Markov game (a subindex i indicates the agent P_i)
$\mathcal{R}_{\mathrm{aver}}$	return model of average reward for a Markov decision process or a two-player zero-sum Markov game
$\mathcal{R}_{\mathrm{disc}}$	return model of discounted reward for a Markov decision process or a two-player zero-sum Markov game
RL	Reinforcement Learning
$\mathcal{S}$	set of states for a Markov decision process or a two-player zero-sum Markov game
$\mathcal{SA}$	set of state action pairs for a Markov decision process
$\mathcal{SAO}$	set of state action pairs for a two-player zero-sum Markov game
S_d	feasible set of a dual linear program
SL	Supervised Learning
S-MDP	Semi Markov Decision Process (see also MDP)
S_p	feasible set of a primal linear program
S_X	permutation or transformation group of a set X containing all automorphisms of X

T	transition probability between two sets of a dynamical system
$T_{\text{inv}}(S)$	invariance ratio of a set S of a dynamical system being the transition probability of a set into itself
$T_{\text{inv}}(\mathcal{P}(S))$	average invariance ratio of a partition of a set S
T	transition function for a Markov decision process or a two-player zero-sum Markov game
$\widetilde{T}$	alternative form of a transition function for a deterministic Markov decision process or a two-player zero-sum Markov game
$U_\epsilon(x)$	ϵ-neighbourhood of $x \subseteq X$
V	vertex set of a graph G
V^π	state value function of policy π for a Markov decision process or a two-player zero-sum Markov game
$V^*(= V^{\pi^*})$	optimal value function for a Markov decision process or a two-player zero-sum Markov game (especially value of a matrix game)
$\widetilde{V}_k$	numerical approximation (e. g. by SL techniques) of the k-th iterate of value iteration
$\mathbb{Z}$	set of integer numbers

Bibliography

[1] Adrian K. Agogino and Kagan Tumer. Unifying temporal and structural credit assignment problems. In *AAMAS*, pages 980–987. IEEE Computer Society, 2004.

[2] James S. Albus. A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *ASM Journal of Dynamic Systems, Measurement, and Control*, 97:220–227, 1975.

[3] James S. Albus. *Brains, Behavior, and Robotics*. Byte Books, Peterborough, New Hampshire, 1981.

[4] Michael A. Arbib, editor. *The Handbook of Brain Theory and Neural Networks*. MIT Press, Cambridge, MA, 1995.

[5] Kenneth J. Arrow, David Blackwell, and M. A. Girshick. Bayes and minimax solutions of sequential decision problems. *Econometrica*, 17:213–244, 1949.

[6] Christopher G. Atkeson and Stefan Schaal. Memory-based neural networks for robot learning. *Neurocomputing*, 9(3):243–269, 1995.

[7] Leemon C. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*, pages 30–37, San Francisco, CA, 1995. Morgan Kaufmann.

[8] Leemon C. Baird and A. Harry Klopf. Reinforcement learning with high-dimensional continuous actions. Technical Report WL-TR-93-1147, Wright Laboratory, Wright-Patterson Air Force Base, OH 45433-7301, 1993.

[9] Martino Bardi, Maurizio Falcone, and Pierpaolo Soravia. Fully discrete schemes for the value function of pursuit-evasion games. In *Advances in dynamic games and applications*, volume 1 of *Annals of the International Society of Dynamic Games*, pages 89–105. Birkhäuser, Boston, MA, 1994.

[10] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.

[11] Andrew G. Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379, 2003.

[12] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuron-like adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(5):834–846, 1983.

[13] Tamer Başar and Geert J. Olsder. *Dynamic Noncooperative Game Theory*. Academic Press Ltd., London, 2nd edition, 1995.

[14] Richard E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[15] Richard E. Bellman and Joseph P. LaSalle. On non-zero sum games and stochastic processes. RM-212, Rand Corp., Santa Monica, 1949.

[16] Hamid R. Berenji. Artificial neural networks and approximate reasoning for intelligent control in space. In *American Control Conference*, pages 1075–1080, 1991.

[17] Donald A. Berry and Bert Fristedt. *Bandit Problems: Sequential Allocation of Experiments*. Chapman and Hall, London, UK, 1985.

[18] Dimitri P. Bertsekas and David A. Castañon. Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transactions on Automatic Control*, 34(6):589–598, 1989.

[19] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[20] Emile Borel. The theory of play and integral equations with skew symmetric kernels. *Econometrica. Journal of the Econometric Society*, 21:97–100, 1953.

[21] Michael Bowling. *Multiagent learning in the presence of agents with limitations*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, 2003. Also published as Technical Report CMU-CS-03-118.

[22] Michael Bowling and Manuela M. Veloso. Existence of multiagent equilibria with limited agents. Technical Report CMU-CS-02-104, Carnegie Mellon University, Pittsburgh, PA, 2002.

[23] Justin A. Boyan and Andrew W. Moore. Generalization in reinforcement learning: Safely approximating the value function. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems (NIPS) 7*, pages 369–376, Cambridge, MA, 1995. The MIT Press.

[24] Glen E. Bredon. *Introduction to Compact Transformation Groups*. Academic Press, New York and London, 1972.

[25] Michael Brin and Garrett Stuck. *Introduction to Dynamical Systems*. Cambridge University Press, 2002.

[26] David N. Burghes and Alexander Graham. *Introduction to Control Theory, Including Optimal Control*. Ellis Horwood Ltd., Chichester, 1980. Ellis Horwood Series in Mathematics and its Applications.

[27] A. Martin V. Butz, David E. Goldberg, and C. Wolfgang Stolzmann. The anticipatory classifier system and genetic generalization. *Natural Computing*, 1(4):427–467, 2002.

[28] Mary L. Cartwright and John E. Littlewood. On non-linear differential equations of the second order. *Journal of the London Mathematical Society. Second Series*, 20:180–189, 1945.

[29] Anthony R. Cassandra, Leslie P. Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the 12th National Conference on Artificial Intelligence*, volume 2, pages 1023–1028, Seattle, WA, 1994. AAAI Press.

[30] Arthur Cayley. Adding temporary memory to ZCS. *The Educational Times*, 23(18), 1875.

[31] David Chapman and Leslie P. Kaelbling. Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In J. Myopoulos and

R. Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI), Sydney, Australia*, pages 726–731, San Francisco, CA, 1991. Morgan Kaufmann.

[32] Bruno Codenotti and Daniel Stefankovic. On the computational complexity of Nash equilibria for $(0, 1)$ bimatrix games. *Information Processing Letters (IPL)*, 94(3):145–150, 2005.

[33] Anne Condon. The complexity of stochastic games. *Information and Computation*, 96(2):203–224, 1992.

[34] Vincent Conitzer and Tuomas Sandholm. Complexity results about Nash equilibria. In Georg Gottlob and Toby Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI), Acapulco, Mexico*, pages 765–771. Morgan Kaufmann, 2003.

[35] Rémi Coulom. *Reinforcement Learning Using Neural Networks, with Applications to Motor Control*. PhD thesis, Institut National Polytechnique de Grenoble, 2002.

[36] Michael G. Crandall. Viscosity solutions: A primer. In *Viscosity solutions and applications*, volume 1660 of *Lecture Notes in Mathematics*, pages 1–43. Springer, Berlin, 1997.

[37] Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society (BAMS)*, 39, 2002.

[38] Shu-Lin Cui, Ji-Gui Sun, Ming-Hao Yin, and Shuai Lu. Solving uncertain Markov decision problems: An interval-based method. In L. Jiao, L. Wang, X. Gao, J. Liu, and F. Wu, editors, *Proceedings of the 2nd International Conference on Advances in Natural Computation (ICNC), Xi'an, China (Part II)*, volume 4222 of *Lecture Notes in Computer Science*, pages 948–957. Springer, 2006.

[39] George B. Dantzig and Mukund N. Thapa. *Linear Programming 2*. Springer Series in Operations Research. Springer-Verlag, New York, 2003.

[40] Konstantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. *Electronic Colloquium on Computational Complexity (ECCC)*, (115), 2005.

[41] Ruchira S. Datta. Universality of Nash equilibria. *Mathematics of Operations Research (MOR)*, 28(3):424–432, 2003.

[42] Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Advances in Neural Information Processing Systems (NIPS) 5*, pages 271–278, San Mateo, CA, 1993. Morgan Kaufmann.

[43] Thomas Dean, Leslie P. Kaelbling, Jak Kirman, and Ann Nicholson. Planning with deadlines in stochastic domains. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI)*, pages 574–579, Washington, DC, 1993. AAAI Press.

[44] Richard Dearden. Structured prioritized sweeping. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, pages 82–89, San Francisco, CA, 2001. Morgan Kaufmann.

[45] Michael Dellnitz, Gary Froyland, and Oliver Junge. The algorithms behind GAIO – set oriented numerical methods for dynamical systems. In *Ergodic Theory, Analysis, and Efficient Simulation of Dynamical Systems*, pages 145–174. Springer, 2001.

[46] Michael Dellnitz, Mirko Hessel-von Molo, Philipp Metzner, Robert Preis, and

Christof Schütte. Graph algorithms for dynamical systems. In A. Mielke, editor, *Analysis, modeling and simulation of multiscale problems*, pages 619–645. Springer, Berlin, 2006.

[47] Michael Dellnitz and Andreas Hohmann. A subdivision algorithm for the computation of unstable manifolds and global attractors. *Numerische Mathematik*, 75(3):293–317, 1997.

[48] Michael Dellnitz, Andreas Hohmann, Oliver Junge, and Martin Rumpf. Exploring invariant sets and invariant measures. *Chaos. An Interdisciplinary Journal of Nonlinear Science*, 7(2):221–228, 1997.

[49] Michael Dellnitz and Oliver Junge. On the approximation of complicated dynamical behavior. *SIAM Journal on Numerical Analysis*, 36(2):491–515 (electronic), 1999.

[50] Michael Dellnitz and Oliver Junge. Set oriented numerical methods for dynamical systems. In *Handbook of dynamical systems, Vol. 2*, pages 221–264. North-Holland, Amsterdam, 2002.

[51] Kan Deng and Andrew W. Moore. Multiresolution instance-based learning. In Chris S. Mellish, editor, *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1233–1242, San Mateo, 1995. Morgan Kaufmann.

[52] Ralf Diekmann, Burkhard Monien, and Robert Preis. Using helpful sets to improve graph bisections. In D. Hsu, A. Rosenberg, and D. Sotteau, editors, *Interconnection Networks and Mapping and Scheduling Parallel Computations*, volume 21 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 57–73. American Mathematical Society, 1995.

[53] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research (JAIR)*, 13:227–303, 2000.

[54] Marco Dorigo and Hugues Bersini. A comparison of Q-learning and classifier systems. In D. Cliff, P. Husbands, J.-A. Meyer, and S. W. Wilson, editors, *From Animals to Animats 3. Proceedings of the 3rd International Conference on Simulation of Adaptive Behavior (SAB)*, pages 248–255, Cambridge, MA, 1994. MIT Press.

[55] Marco Dorigo and Marco Colombetti. Robot shaping: Developing autonomous agents through learning. *Artificial Intelligence*, 71(2):321–370, 1994.

[56] Kenji Doya. Reinforcement learning in continuous time and space. *Neural Computation*, 12(1):219–245, 2000.

[57] Jan Drugowitsch and Alwyn M. Barry. A formal framework and extensions for function approximation in learning classifier systems. Technical Report CSBU-2006-02, University of Bath, 2006.

[58] Chris Drummond. Accelerating reinforcement learning by composing solutions of automatically identified subtasks. *Journal of Artificial Intelligence Research (JAIR)*, 16:59–104, 2002.

[59] Artur Dubrawski and Jeff G. Schneider. Memory based stochastic optimization for validation and tuning of function approximators. In *Proceedings of the 6th International Workshop on AI and Statistics, Florida, USA*, 1997.

[60] David S. Dummit and Richard M. Foote. *Abstract algebra*. Prentice Hall Inc., En-

glewood Cliffs, NJ, 1991.

[61] Aryeh Dvoretzky, J. Kiefer, and Jacob Wolfowitz. The inventory problem. I. Case of known distributions of demand. *Econometrica*, 20:187–222, 1952.

[62] Scott E. Fahlman. An empirical study of learning speed in back-propagation networks. Technical Report CMU-CS-88-162, Carnegie-Mellon University, Pittsburgh, PA, 1988.

[63] Maurizio Falcone. Numerical methods for differential games based on partial differential equations. Unpublished. Based on lectures given at the summer school on Differential Games and Applications, 2005.

[64] Jacques Ferber. *Multi-Agent Systems – An Introduction to Distributed Artifical Intelligence*. Addison Wesley, 1999.

[65] Jerzy A. Filar, T. A. Schultz, Frank Thuijsman, and Koos (O. J.) Vrieze. Nonlinear programming and stationary equilibria in stochastic games. *Mathematical Programming*, 50(2):227–237, 1991.

[66] Jerzy A. Filar and Koos (O. J.) Vrieze. *Competitive Markov Decision Processes*. Springer-Verlag, New York, 1997.

[67] David Foster and Peter Dayan. Structure in the space of value functions. *Machine Learning*, 49(2–3):325–346, 2002.

[68] Bas van Fraassen. *Laws and Symmetry*. Oxford University Press, Oxford, 1989.

[69] Thomas Gabel, Roland Hafner, Sascha Lange, Martin Lauer, and Martin Riedmiller. Bridging the gap: Learning in the robocup simulation and midsize league. In *Proceedings of the 7th Portuguese Conference on Automatic Control (Controlo), Lisbon, Portugal*, 2006.

[70] Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147(1–2):163–223, 2003.

[71] David E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[72] Geoffrey J. Gordon. Stable function approximation in dynamic programming. In A. Prieditis and S. Russell, editors, *Proceedings of the 12th International Conference on Machine Learning (ICML)*, pages 261–268, San Francisco, CA, 1995. Morgan Kaufmann.

[73] John Guckenheimer and Philip Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*. Springer-Verlag, New York, 1990.

[74] Carlos Guestrin, Milos Hauskrecht, and Branislav Kveton. Solving factored MDPs with continuous and discrete variables. In D. M. Chickering and J. Y. Halpern, editors, *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI), Banff, Canada*, pages 235–242, 2004.

[75] Carlos Guestrin, Daphne Koller, and Ronald Parr. Max-norm projections for factored MDPs. In B. Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01), Seattle*, pages 673–682, San Francisco, CA, 2001. Morgan Kaufmann.

[76] Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored MDPs. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in*

Neural Information Processing Systems (NIPS) 14, Vancouver, Canada, pages 1523–1530, Cambridge, MA, 2001. MIT Press.

[77] Carlos Guestrin, Daphne Koller, and Ronald Parr. Solving factored POMDPs with linear value functions. In *Workshop on Planning under Uncertainty and Incomplete Information (IJCAI), Seattle, Washington*, pages 67–75, 2001.

[78] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research (JAIR)*, 19:399–468, 2003.

[79] Carlos Guestrin, Michail G. Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In C. Sammut and A. G. Hoffmann, editors, *Proceedings of the 19th International Conference on Machine Learning (ICML), Sydney, Australia*, pages 227–234, San Francisco, CA, 2002. Morgan Kaufmann.

[80] Vijaykumar Gullapalli. *Reinforcement Learning and its Application to Control*. PhD thesis, University of Massachusetts at Amherst, 1992.

[81] Milos Hauskrecht, Nicolas Meuleau, Leslie P. Kaelbling, Thomas Dean, and Craig Boutilier. Hierarchical solution of Markov decision processes using macro-actions. In G. F. Cooper and S. Moral, editors, *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 220–229, San Francisco, 1998. Morgan Kaufmann.

[82] Simon S. Haykin. *Neural Networks: A Comprehensive Introduction*. Prentice Hall, New Jersey, USA, 1999.

[83] Chao He, Li-Xin Xu, and Yu-He Zhang. Learning convergence of CMAC algorithm. *Neural Processing Letters*, 14(1):61–74, 2001.

[84] Ronald A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

[85] Fern Y. Hunt. A Monte Carlo approach to the approximation of invariant measures. *Random & Computational Dynamics*, 2(1):111–133, 1994.

[86] Rufus P. Isaacs. *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. John Wiley, Toronto, 1965.

[87] Tommi Jaakkola, Satinder P. Singh, and Michael I. Jordan. Reinforcement learning algorithm for partially observable Markov decision problems. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems (NIPS 7)*, pages 345–352, Cambridge, MA, 1995. The MIT Press.

[88] Leslie P. Kaelbling, Michael L. Littman, and Andrew P. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research (JAIR)*, 4:237–285, 1996.

[89] Spiros Kapetanakis and Daniel Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *AAAI/IAAI 2002*, pages 326–331, 2002.

[90] Spiros Kapetanakis, Daniel Kudenko, and Malcolm J. A. Strens. Learning of coordination in cooperative multi-agent systems using commitment sequences. In *Artificial Intelligence and the Simulation of Behavior 1(5)*, 2004.

[91] George Karypis and Vipin Kumar. *METIS Manual, Version 4.0*. University of Minnesota, 1998.

[92] Brian W. Kernighan and Shen Lin. An efficient heuristic procedure for partitioning

graphs. *Bell System Technical Journal*, 49(2):291–307, 1970.

[93] Teuvo Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 1995.

[94] Daphne Koller and Ronald Parr. Computing factored value functions for policies in structured MDPs. In T. Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1332–1339, San Francisco, CA, 1999. Morgan Kaufmann.

[95] Panganamala R. Kumar and Pravin P. Varaiya. *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Prentice Hall, Englewood Cliffs, NJ, 1986.

[96] Rainer Lachner, Michael H. Breitner, and Hans J. Pesch. Real-time collision avoidance against wrong drivers: Differential game approach, numerical solution and synthesis of strategies with neural networks. In *Proceedings of the 7th International Symposium on Dynamic Games and Applications, Kanagawa, Japan*, 1996.

[97] Michail G. Lagoudakis and Ronald Parr. Value function approximation in zero-sum Markov games. In A. Darwiche and N. Friedman, editors, *Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence (UAI), University of Alberta, Edmonton, Alberta, Canada*, pages 283–292, San Francisco, CA, 2002. Morgan Kaufmann.

[98] Michail G. Lagoudakis and Ronald Parr. Learning in zero-sum team Markov games using factored value functions. In S. Becker, S. B. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS) 15*, pages 1627–1634. MIT Press, Cambridge, MA, 2003.

[99] Michail G. Lagoudakis, Ronald Parr, and Michael L. Littman. Least-squares methods in reinforcement learning for control. In I. P. Vlahavas and C. D. Spyropoulos, editors, *Proceedings of the 2nd Hellenic Conference on AI (SETN). Thessaloniki, Greece*, volume 2308 of *Lecture Notes in Computer Science*, pages 249–260. Springer, 2002.

[100] Pier L. Lanzi. Learning classifier systems from a reinforcement learning perspective. *Soft Computing*, 6(3–4):162–170, 2002.

[101] Tim Laue and Thomas Röfer. Integrating simple unreliable perceptions for accurate robot modeling in the four-legged league. In G. Lakemeyer, E. Sklar, D. G. Sorrenti, and T. Takahashi, editors, *RoboCup*, volume 4434 of *Lecture Notes in Computer Science*, pages 474–482. Springer, 2006.

[102] Martin Lauer and Martin Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proc. of the 17th International Conference on Machine Learning (ICML)*, pages 535–542, San Francisco, CA, 2000. Morgan Kaufmann.

[103] Martin Lauer and Martin Riedmiller. Reinforcement learning for stochastic cooperative multi-agent systems. In *AAMAS*, pages 1516–1517. IEEE Computer Society, 2004.

[104] Steven M. LaValle. Robot motion planning: A game-theoretic foundation. *Algorithmica*, 26(3-4):430–465, 2000.

[105] Chuen-Chien Lee. A self learning rule-based controller employing approximate reasoning and neural net concepts. *International Journal of Intelligent Systems*, 6(1):71–93, 1991.

[106] Carlton E. Lemke and Joseph T. Howson, Jr. Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics (SIAM)*, 12(2):413–423, 1964.

[107] Joseph Lewin. *Differential Games*. Springer-Verlag London Ltd., London, 1994.

[108] Long-Ji Lin. Programming robots using reinforcement learning and teaching. In T. L. Dean and K. McKeown, editors, *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 781–786. MIT Press, 1991.

[109] Long-Ji Lin. Hierarchical learning of robot skills by reinforcement. In *Proceedings of the International Conference on Neural Networks (ICNN)*, volume 1, pages 181–186, San Francisco, CA, 1993. IEEE/INNS.

[110] Long-Ji Lin and Tom M. Mitchell. Memory approaches to reinforcement learning in non-Markovian domains. Technical Report CMU-CS-92-138, Carnegie Mellon University, Pittsburgh, PA, 1992.

[111] Ya-Ping Lin and Xue-Yong Li. Reinforcement learning based on local state feature learning and policy adjustment. *Information Sciences ISCI*, 154(1–2):59–70, 2003.

[112] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ICML)*, pages 157–163, San Francisco, CA, 1994. Morgan Kaufmann.

[113] Michael L. Littman. Memoryless policies: Theoretical limitations and practical results. In D. Cliff, P. Husbands, J.-A. Meyer, and S. W. Wilson, editors, *From Animals to Animats 3: Proceedings of the 3rd International Conference on Simulation of Adaptive Behavior (SAB)*, Cambridge, MA, 1994. MIT Press.

[114] Michael L. Littman, Anthony R. Cassandra, and Leslie P. Kaelbling. Learning policies for partially observable environments: Scaling up. In A. Prieditis and S. Russell, editors, *Proceedings of the 12th International Conference on Machine Learning (ICML)*, pages 362–370, San Francisco, CA, 1995. Morgan Kaufmann Publishers.

[115] Michael L. Littman, Thomas L. Dean, and Leslie P. Kaelbling. On the complexity of solving Markov decision problems. In P. Besnard and S. Hanks, editors, *Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 394–402, San Francisco, CA, 1995. Morgan Kaufmann.

[116] Michael L. Littman and Csaba Szepesvári. A generalized reinforcement-learning model: Convergence and applications. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning (ICML), Bari, Italy*, pages 310–318. Morgan Kaufmann, 1996.

[117] Edward N. Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Science*, 20:130–141, 1963.

[118] Ulf Lorenz and Burkhard Monien. Error analysis in minimax trees. *Theoretical Computer Science (TCS)*, 313(3):485–498, 2004. Algorithmic combinatorial game theory.

[119] William S. Lovejoy. A survey of algorithmic methods for partially observable Markov decision processes. *Annals of Operations Research*, 28(1):47–65, 1991.

[120] R. Duncan Luce and Howard Raiffa. *Games and Decisions: Introduction and Critical Survey*. John Wiley, New York, 1957. A study of the Behavioral Models Project, Bureau of Applied Social Research, Columbia University;.

[121] David J. C. MacKay. Bayesian model comparison and backprop nets. In J. E. Moody, S. J. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems (NIPS) 4*, pages 839–846. Morgan Kaufmann, 1992.

[122] David J. C. MacKay. Bayesian non-linear modelling for the prediction competition. In *ASHRAE Transactions*, volume 100, pages 1053–1062, Atlanta, Georgia, 1994. ASHRAE.

[123] Omid Madani. On policy iteration as a Newton's method and polynomial policy iteration algorithms. In *Proceedings of the 18th National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, pages 273–278, Menlo Parc, CA, 2002. AAAI Press.

[124] Pattie Maes and Rodney A. Brooks. Learning to coordinate behaviors. In T. G. Dietterich and W. Swartout, editors, *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI)*, pages 796–802, Boston, MA, 1990. MIT Press.

[125] Sridhar Mahadevan. To discount or not to discount in reinforcement learning: A case study comparing R learning and Q learning. In *Proceedings of the 11th International Conference on Machine Learning (ICML)*, pages 164–172, San Francisco, CA, 1994. Morgan Kaufmann.

[126] Sridhar Mahadevan. Proto-value functions: Developmental reinforcement learning. In Luc De Raedt and Stefan Wrobel, editors, *Proceedings of the 22nd International Conference on Machine Learning (ICML), Bonn, Germany*, pages 553–560. ACM, 2005.

[127] Sridhar Mahadevan and Jonathan Connell. Scaling reinforcement learning to robotics by exploiting the subsumption architecture. In *Proceedings of the 8th International Workshop on Machine Learning (ICML)*, pages 328–332, 1991.

[128] Sridhar Mahadevan, Mauro Maggioni, Kimberly Ferguson, and Sarah Osentoski. Learning representation and control in continuous Markov decision processes. In *AAAI*. Boston, 2006.

[129] Olvi L. Mangasarian and H. Stone. Two-person nonzero-sum games and quadratic programming. *Journal of Mathematical Analysis and Applications*, 9:348–355, 1964.

[130] Shie Mannor, Ishai Menache, Amit Hoze, and Uri Klein. Dynamic abstraction in reinforcement learning via clustering. In C. E. Brodley, editor, *Proceedings of the 21st International Conference on Machine Learning (ICML), Banff, Canada*. ACM, 2004.

[131] Oded Maron and Andrew W. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems (NIPS) 6*, pages 59–66, San Francisco, CA, 1994. Morgan Kaufmann.

[132] Oded Maron and Andrew W. Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Rev*, 11(1-5):193–225, 1997.

[133] Maja J. Mataric. Reward functions for accelerated learning. In *Proceedings of the 11nd International Conference on Machine Learning (ICML)*, pages 181–189, 1994.

[134] R. Andrew K. McCallum. Instance-based utile distinctions for reinforcement learning with hidden state. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*, pages 387–395, San Francisco, CA, 1995. Morgan Kaufmann.

[135] Richard D. McKelvey and Andrew McLennan. Computation of equilibria in finite games. In *Handbook of Computational Economics, Vol. I*, volume 13 of *Handbooks in Economics*, pages 87–142. North-Holland, Amsterdam, 1996.

[136] John C. C. McKinsey. *Introduction to the Theory of Games*. McGraw-Hill Book Company, Inc., New York-Toronto-London, 1952.

[137] Lisa Meeden, Gary Mcgraw, and Douglas Blank. Emergent control and planning in an autonomous vehicle. In D. S. Touretsky, editor, *Proceedings of the 15th Annual Meeting of the Cognitive Science Society*, pages 735–740. Lawrence Erlbaum, Hillsdale, NJ, 1993.

[138] José del R. Millán. Rapid, safe, and incremental learning of navigation strategies. In *IEEE Transactions on Systems, Man and Cybernetics (Part B)*, volume 26, pages 408–420, 1996.

[139] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[140] George E. Monahan. A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.

[141] Burkhard Monien, Robert Preis, and Ralf Diekmann. Quality matching and local improvement for multilevel graph-partitioning. *Parallel Computing*, 26(12):1609–1634, 2000.

[142] Andrew W. Moore. Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In L. Birnbaum and G. Collins, editors, *Proceedings of the 8th International Conference on Machine Learning (ICML)*, pages 333–337, San Francisco, CA, 1991. Morgan Kaufmann.

[143] Andrew W. Moore. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems (NIPS) 6*, pages 711–718, San Mateo, CA, 1994. Morgan Kaufmann.

[144] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103–130, 1993.

[145] Andrew W. Moore, Christopher G. Atkeson, and Stefan Schaal. Memory-based learning for control. Technical Report CMU-RI-TR-95-18, Carnegie Mellon University, Pittsburgh, PA, 1995.

[146] Andrew W. Moore, Daniel J. Hill, and Michael P. Johnson. An empirical investigation of brute force to choose features, smoothers and function approximators. In S. Hanson, S. Judd, and T. Petsche, editors, *Computational Learning Theory and Natural Learning*, volume III: Selecting Good Models, pages 361–379. MIT Press, 1995.

[147] Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural Computation*, 17(2):335–359, 2005.

[148] Rémi Munos. A study of reinforcement learning in the continuous case by the means of viscosity solutions. *Machine Learning*, 40(3):265–299, 2000.

[149] Rémi Munos. Error bounds for approximate policy iteration. In T. Fawcett and N. Mishra, editors, *Proceedings of the 20th International Conference on Machine Learning (ICML), Washington, DC, USA*, pages 560–567. AAAI Press, 2003.

[150] Rémi Munos. Policy gradient in continuous time. *Journal of Machine Learning*

Research (JMLR), 7:771–791, 2006.

[151] Rémi Munos. Performance bounds in l_p norm for approximate value iteration. *SIAM Journal on Control and Optimization*, 2007.

[152] Rémi Munos, Leemon C. Baird, and Andrew W. Moore. Gradient descent approaches to neural-net-based solutions of the Hamilton-Jacobi-Bellman equation. In *International Joint Conference on Neural Networks (IJCNN)*, volume 3, pages 2152–2157, 1999.

[153] Ali H. Nayfeh and Balakumar Balachandran. *Applied Nonlinear Dynamics*. Wiley Series in Nonlinear Science. John Wiley, New York, 1995.

[154] John von Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematical Annals*, 100:295–320, 1928.

[155] John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ, 2nd edition, 1947.

[156] Ralph Neuneier and Hans G. Zimmermann. How to train neural networks. In G. B. Orr and K. R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pages 373–423. Springer, 1996.

[157] Partha Niyogi and Federico Girosi. Generalization bounds for function approximation from scattered noisy data. *Advances in Computational Mathematics*, 10:51–80, 1999.

[158] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, Cambridge, MA, 1994.

[159] Guillermo Owen. *Game Theory*. Academic Press Inc., San Diego, CA, 3rd edition, 1995.

[160] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[161] Ronald E. Parr. *Hierarchical Control and Learning for Markov Decision Processes*. PhD thesis, University of California, Berkeley, 1998.

[162] Relu Patrascu, Pascal Poupart, Dale Schuurmans, Craig Boutilier, and Carlos Guestrin. Greedy linear value-approximation for factored Markov decision processes. In *Proceedings of the 18th National Conference on Artificial Intelligence and 14th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, pages 285–291, Menlo Parc, CA, 2002. AAAI Press.

[163] Jing Peng and Ronald J. Williams. Efficient learning and planning within the Dyna framework. *Adaptive Behavior*, 1(4):437–454, 1993.

[164] Jing Peng and Ronald J. Williams. Incremental multi-step Q-learning. In *Proceedings of the 11th International Conference on Machine Learning (ICML)*, pages 226–232, San Francisco, CA, 1994. Morgan Kaufmann.

[165] Hans J. Pesch, I. Gabler, Stefan Miesbach, and Michael H. Breitner. Synthesis of optimal strategies for differential games by neural networks. In G. J. Olsder, editor, *New Trends in Dynamic Games and Applications*, Annals of the International Society of Dynamic Games 3, pages 111–142, Boston, 1996. Birkhäuser.

[166] Robert Preis. *The PARTY Graphpartitioning-Library, User Manual – Version 1.99*. University of Paderborn, 1998.

[167] Bob Price and Craig Boutilier. Accelerating reinforcement learning through implicit

imitation. *Journal of Artificial Intelligence Research (JAIR)*, 19:569–629, 2003.

[168] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley, New York, 1994.

[169] Jette Randløv. *Solving Complex Problems with Reinforcement Learning*. PhD thesis, University of Copenhagen, 2001.

[170] Balaraman Ravindran and Andrew G. Barto. Symmetries and model minimization in Markov decision processes. Technical Report CMPSCI 01-43, University of Massachusetts, Amherst, MA, 2001.

[171] Balaraman Ravindran and Andrew G. Barto. Model minimization in hierarchical reinforcement learning. In S. Koenig and R. C. Holte, editors, *5th International Symposium on Abstraction, Reformulation and Approximation (SARA), Kananaskis, Canada*, volume 2371 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 2002.

[172] Mark B. Ring. *Continual Learning in Reinforcement Environments*. PhD thesis, University of Texas at Austin, 1994.

[173] Nicholas Roy. *Finding Approximate POMDP Solutions through Belief Compression*. PhD thesis, Carnegie Mellon University, 2003. Also published as Technical Report CMU-RI-TR-03-25.

[174] Nicholas Roy, Geoffrey J. Gordon, and Sebastian B. Thrun. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research (JAIR)*, 23:1–40, 2005.

[175] Ulrich Rüde. *Mathematical and Computational Techniques for Multilevel Adaptive Methods*. SIAM, Philadelphia, PA, 1993.

[176] David E. Rumelhart and James L. McClelland. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.*, volume 1–2. MIT Press, Cambridge, MA, 1986.

[177] Gavin A. Rummery. *Problem Solving with Reinforcement Learning*. PhD thesis, University of Cambridge, 1995.

[178] Rafal P. Salustowicz, Marco A. Wiering, and Jürgen Schmidhuber. Learning team strategies: Soccer case studies. *Machine Learning*, 33:263–282, 1998.

[179] Arthur L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3:211–229, 1959. Reprinted in E. A. Feigenbaum and J. Feldman, editors, *Computers and Thought*, McGraw-Hill, NY 1963.

[180] Arthur L. Samuel. Some studies in machine learning using the game of checkers II – Recent progress. *IBM Journal of Research and Development*, 11(6):601–617, 1967.

[181] Uday Savagaonkar, Edwin K. P. Chong, and Robert L. Givan. Sampling techniques for zero-sum, discounted Markov games. In Leslie P. Kaelbling, editor, *Allerton Conference on Control and Communications*, 2002.

[182] Jürgen Schmidhuber. Reinforcement learning in Markovian and non-Markovian environments. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems (NIPS) 3*, pages 500–506. Morgan Kaufmann, 1991.

[183] Jürgen Schmidhuber. Developmental robotics, optimal artificial curiosity, creativity,

music, and the fine arts. In *Connection Science*, volume 18, pages 173–187, 2006.

[184] Alexander Schrijver. *Theory of Linear and Integer Programming*. Wiley-Interscience Series in Discrete Mathematics. John Wiley, 1986.

[185] Anton Schwartz. A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the 10th International Conference on Machine Learning (ICML)*, San Mateo, CA, 1993. Morgan Kaufmann.

[186] Lloyd S. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences of the U. S. A.*, 39:1095–1100, 1953.

[187] John W. Sheppard. Co-learning in differential games. *Machine Learning*, 33(2-3):201–233, 1998.

[188] Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? In R. Vohra and M. Wellman, editors, *Artificial Intelligence (special issue on foundations of multi-agent learning)*, volume 171, pages 365–377, 2007.

[189] Sajid M. Siddiqi and Andrew W. Moore. Fast inference and learning in large-state-space HMMs. In L. de Raedt and S. Wrobel, editors, *Proceedings of the 22nd International Conference on Machine Learning (ICML), Bonn, Germany*, pages 800–807. ACM, 2005.

[190] Satinder P. Singh. Reinforcement learning with a hierarchy of abstract models. In W. R. Swartout, editor, *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI), San Jose, CA*, pages 202–207. MIT Press, 1992.

[191] Margaret M. Skelly. *Hierarchical Reinforcement Learning with Function Approximation for Adaptive Control*. PhD thesis, Case Western Reserve University, OhioLINK, 2004.

[192] Stephen Smale. Differentiable dynamical systems. *Bulletin of the American Mathematical Society (BAMS)*, 73:747–817, 1967.

[193] Andrew J. Smith. Applications of the self-organising map to reinforcement learning. *Neural Networks*, 15(8-9):1107–1124, 2002.

[194] Bernhard von Stengel. Computing equilibria for two-person games. In R. J. Aumann and S. Hart, editors, *Handbook of Game Theory with Economic Applications*, volume 3, chapter 45. North-Holland, Amsterdam, 2002.

[195] Robert F. Stengel. *Stochastic Optimal Control*. A Wiley-Interscience Publication. John Wiley, New York, 1986.

[196] Eduard L. Stiefel. Note on Jordan elimination, linear programming and Chebyshev approximation. *Numerische Mathematik*, 2:1–17, 1960.

[197] Peter Stone. *Layered Learning in Multi-Agent Systems*. PhD thesis, Carnegie Mellon University, 1998.

[198] Peter Stone and Richard S. Sutton. Scaling reinforcement learning toward robocup soccer. In C. E. Brodley and A. P. Danyluk, editors, *Proceedings of the 18th International Conference on Machine Learning (ICML), Williams College, Williamstown, MA*, pages 537–544, San Francisco, CA, 2001. Morgan Kaufmann.

[199] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning (ICML)*, pages 216–224, Austin, TX, 1990. Morgan

Kaufmann.

[200] Richard S. Sutton. Planning by incremental dynamic programming. In *Proceedings of the 8th International Workshop on Machine Learning (ICML)*, pages 353–357. Morgan Kaufmann, 1991.

[201] Richard S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D. S. Touretzky, M. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems (NIPS) 8*, pages 1038–1044, Cambridge, MA, 1996. MIT Press.

[202] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

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

[204] Erik Talvitie and Satinder P. Singh. An experts algorithm for transfer learning. In M. M. Veloso, editor, *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI), Hyderabad, India*, pages 1065–1070, 2007.

[205] Gerald Tesauro. TD-Gammon, A self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

[206] Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[207] Sebastian B. Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In M. Mozer, P. Smolensky, D. Touretzky, J. Elman, and A. Weigend, editors, *Proceedings of the 1993 Connectionist Models Summer School*, Hillsdale, NJ, 1993. Lawrence Erlbaum.

[208] Michael J. Todd. The many facets of linear programming. *Mathematical Programming*, 91(3):417–436, 2002.

[209] Abraham Wald. Generalization of a theorem by v. Neumann concerning zero sum two person games. *Annals of Mathematics. Second Series*, 46:281–286, 1945.

[210] Abraham Wald. *Sequential Analysis*. John Wiley, New York, 1947.

[211] Chris Walshaw. *The JOSTLE user manual: Version 2.2*. University of Greenwich, 2000.

[212] Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, University of Cambridge, 1989.

[213] Gerhard Weiß. A multiagent framework for planning, reacting, and learning. Technical Report FKI-233-99, TU München, Germany, 1999.

[214] Jochen Werner. *Numerische Mathematik*. Vieweg Verlag, Braunschweig, Germany, 1992.

[215] Shimon Whiteson and Peter Stone. Concurrent layered learning. In *Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 193–200. ACM, 2003.

[216] Chang Zhang and John S. Baras. A new adaptive aggregation algorithm for infinite horizon dynamic programming. In *Proceedings of the 11th Mediterranean Conference on Control and Automation (MED), Rhodes, Greece*, 2003.

[217] Martin Zinkevich and Tucker Balch. Symmetry in Markov decision processes and its

implications for single agent and multiagent learning. In *Proceedings of the 18th International Conference on Machine Learning (ICML), Williams College, Williamstown, MA*, pages 632–639, San Francisco, CA, 2001. Morgan Kaufmann.

[218] Martin Zinkevich, Amy Greenwald, and Michael G. Littman. Cyclic equilibria in Markov games. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems (NIPS) 18*, pages 1641–1648. MIT Press, Cambridge, MA, 2006.

Index