

Sport-Strategy Optimization with Markov Decision Processes

Von der Universität Bayreuth
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

von

Susanne Hoffmeister

aus Groß-Umstadt

1. Gutachter Prof. Dr. Jörg Rambau
2. Gutachter Prof. Dr. Gautier Stauffer

Tag der Einreichung: 19. Dezember 2018
Tag des Kolloquiums: 05. April 2019

Zusammenfassung

Sport-Strategie Optimierung behandelt strategische Fragestellungen im Sport, die von Trainern oder Spielern getroffen werden. Durch die steigende Menge an erfassten Daten während eines Sportspiels, steigt auch der Bedarf an einer datengetriebenen Entscheidungshilfe. Müssen zwei gegenläufige Effekte abgewogen werden, so kann ein datengetriebener Ansatz neue Erkenntnisse erzielen, die durch eine rein qualitative Betrachtung nicht möglich wären.

Das zu Grunde liegende mathematische Werkzeug dieser Thesis sind *Markov Decision Problems (MDPs)*. Die Arbeit beinhaltet eine theoretische Analyse geeigneter MDPs, ihre Anwendung auf Beachvolleyball und einen neuen Ansatz, der zwei MDPs mit unterschiedlichem Detaillierungsgrad kombiniert.

Es wird eine neue Klasse namens *Sport-Strategy Optimization MDPs (SSO-MDPs)* eingeführt, die sich zur Modellierung von Sportspielen eignet. SSO-MDPs maximieren die Gewinn-Wahrscheinlichkeit eines Spiels, während jede Aktion eine positive Fehlerwahrscheinlichkeit besitzt. In einer theoretischen Analyse von SSO-MDPs wird bewiesen, dass die *Optimality Equations* für SSO-MDPs einen eindeutigen Fixpunkt haben und der *Dynamic Programming Operator* angewandt auf SSO-MDPs eine Kontraktionsabbildung definiert. Darüber hinaus wird ein lineares Programm für SSO-MDPs hergeleitet.

Diese Arbeit beinhaltet zwei SSO-MDPs, die eine sportstrategische Frage im Beachvolleyball modellieren. Beide Modelle haben sehr unterschiedliche Detaillierungsgrade, die zu modellspezifischen Vor- und Nachteilen führen. In einem neuen Ansatz namens *Two-Scale Approach (2-MDP approach)* werden beiden SSO-MDPs kombiniert, um deren Nachteile zu überwinden.

Die einzelnen SSO-MDPs und der 2-MDP approach werden am Beispiel des Beachvolleyballfinals im Rahmen der Olympischen Spiele 2012 in London ausgewertet. Für diesen Zweck wurden umfangreiche Daten aus Videoaufzeichnungen erhoben. Die eingangs gestellten spielstrategischen Fragen werden sowohl durch Auswertung der einzelnen SSO-MDPs und als auch mit Hilfe des 2-MDP approach beantwortet. Die tatsächlichen Realisierungen im Finale werden zur Validierung der Modelle herangezogen.

Basierend auf dem 2-MDP approach werden zwei Werkzeuge für Trainer und Spieler entwickelt. *Strategy-Skill Score Cards* vereinen zwei Sensitivitätsanalysen in einem Diagramm. Anhand des Diagramms können Spielsituationen identifiziert werden, bei denen die optimale Strategie von kleinen Unterschieden in der Tagesform abhängt. Das zweite Werkzeug ist eine farbkodierte Tabelle eines *constant-sum matrix-games*, mit der vielversprechende Strategiemuster identifiziert werden können.

Abstract

Sport-strategy optimization deals with strategic questions in sports games that are made by coaches and players. As the amount of collected data from sports games increases, the need for a data-driven decision support increases, too. Especially if opposing effects need to be weighed up, a data-driven approach can uncover insights that are not available from a solely qualitative analysis.

The underlying mathematical framework used in this thesis are *Markov decision problems (MDPs)*. The thesis is divided up in: a theoretical analysis of suitable MDPs, their application to beach volleyball and a new approach that combines two MDPs with different granularity.

A new class of MDPs suitable to model sports games is introduced and called *Sport-Strategy Optimization MDPs (SSO-MDPs)*. SSO-MDPs maximize the probability of winning a match while every action has a positive probability to fail. A theoretical analysis of SSO-MDPs proves that the optimality equations of SSO-MDPs have a unique fixed point and the dynamic programming operator applied to SSO-MDPs is a contraction mapping. Furthermore, a linear programming formulation for SSO-MDPs is deduced.

This thesis includes two SSO-MDPs that model the same sport-strategic question in beach volleyball. The two models have different levels of detail which lead to different advantages and disadvantages. To overcome the downsides of the individual models, an approach called *Two-Scale Approach (2-MDP approach)* is introduced that combines two SSO-MDPs.

The single SSO-MDPs and the 2-MDP approach are evaluated and compared on real data sets that were collected from the beach volleyball final of the Olympic games 2012 in London. The realizations in the final match are used for validating the individual SSO-MDPs and the 2-MDP approach.

Two tools for coaches and players are developed based on the 2-MDP approach. A *Strategy-Skill Score Card* combines two sensitivity analyses in one diagram. These diagrams can be used to identify critical situations where the optimal strategy is affected by small differences of the player's performance. The other tool is a table of a *constant-sum matrix-game* that can be used to identify promising strategy patterns.

Contents

- 1 Introduction to Sport-Strategy Optimization ii

- 2 Theory of Markov Decision Problems 15
 - 2.1 Basic Definition of Markov Decision Problems (MDPs) 16
 - 2.1.1 Markov Decision Processes 16
 - 2.1.2 Optimality Criteria 19
 - 2.2 Finite Horizon MDPs 21
 - 2.2.1 Policy Evaluation 22
 - 2.2.2 Optimality Equations and Backward Induction 23
 - 2.3 Infinite-Horizon Expected Total Reward MDPs 24
 - 2.3.1 Classification 25
 - 2.3.2 Vector Notation 36
 - 2.3.3 Optimality Equations 36
 - 2.3.4 Solutions of the Optimality Equations 38
 - 2.3.5 Value Iteration 43
 - 2.3.6 Policy Iteration 44
 - 2.3.7 Linear Programming Formulation 49
 - 2.4 Graph Theory and Maximum Flow Problems 61
 - 2.5 Relation to Markov Games 66

- 3 MDPs for Sport-Strategy Optimization 71
 - 3.1 Introduction to MDPs in Sports Games 71
 - 3.2 Definition of Sport-Strategy Optimization MDPs (SSO-MDPs) 75
 - 3.3 Classification 79
 - 3.4 Theoretical Analysis 82
 - 3.5 Linear Programming Formulations 86
 - 3.5.1 A Primal Linear Programming Formulation 87
 - 3.5.2 A Dual Linear Programming Formulation 90
 - 3.6 Flow Networks associated with SSO-MDPs 93
 - 3.6.1 Basic Definitions 94
 - 3.6.2 Induced Flow of an SSO-MDP 96
 - 3.6.3 Maximum Flow Problem for SSO-MDPs 108
 - 3.7 Transforming SSO-MDPs 136

3.8	Further Extensions of SSO-MDPs	149
3.8.1	Randomized Strategies	149
3.8.2	Extension to Markov Games	150
4	Application to Beach Volleyball	155
4.1	Introduction to Beach Volleyball	155
4.1.1	Literature Overview – Modeling Return Plays	155
4.1.2	Summary of Beach Volleyball Rules	156
4.2	An SSO-MDP for a Beach Volleyball Set	158
4.2.1	Definition	158
4.2.2	Transformation	163
4.2.3	Mathematical Analysis	166
4.2.4	Winning Probability of the Tie-Game	168
4.2.5	Application to a Match	169
4.3	An SSO-MDP for a Beach Volleyball Rally	173
4.3.1	Definition	174
4.3.2	Defining a Decision Rule	187
4.3.3	Application to a Match	193
4.3.4	Solving the Rally-SSO-MDP	204
5	A Two-Scale Approach	209
5.1	General Procedure	209
5.1.1	Sport-Strategic Question	209
5.1.2	Modeling Granularity	211
5.1.3	The Underlying Idea	213
5.1.4	Formalization	214
5.2	A Two-Scale Approach for Beach Volleyball	217
5.2.1	Implementation for Beach Volleyball	218
5.2.2	Validating the Implementation	220
5.2.3	Answering Strategic Questions	222
5.2.4	Comparison of Results	223
5.2.5	Sensitivity Analysis	225
5.2.6	Two-Person-Constant-Sum Game	232
6	Conclusion and Outlook	237
A	Rally-SSO-MDP: Skill Estimates	241
A.1	Brink	241
A.2	Reckermann	243
A.3	Alison	245
A.4	Emanuel	247
B	Rally-SSO-MDP: Simulation	251

C	Rally-SSO-MDP: Basic Decision Rule	255
C.1	Definition of the Basic Decision Rule	255
C.1.1	Serving States	256
C.1.2	Reception States	258
C.1.3	Defense States	261
C.1.4	Setting States	263
C.1.5	Attacking States	267
D	Set-SSO-MDP: Strategy Estimates of Team Q	271
E	Data and Software	273

Chapter 1

Introduction to Sport-Strategy Optimization

Sport-Strategy optimization supports strategic decisions in sports games that concern a particular team facing a specific opponent in an upcoming match.

A (team-)strategy makes a set of individual players to become a real team. Furthermore, a superior strategy can lead the technical inferior team or player to win. For example, Terroba et al. considered in Terroba et al. (2013) the 2010 Australian Open Women's Semi Final between Na Li and Serena Williams and found a strategy with which Na Li could have beaten Serena Williams. It is the coaches' and team leaders' job to determine the strategy to play in a match. Those experts have a vast wealth of experience. However, in some situation where it is not clear which strategy outperforms the other strategies, data-driven decision support may help the coaches to make their decision. In these situations sport-strategy optimization can make a difference: It can support the coaches to make their decisions, it can be a tool to justify those decisions, it can quantify how large the gap between several strategies is, and it can give the coach a hint to consider a completely different strategy. However, sport-strategy optimization is only meant to be a support for the coaches. It should not replace the decision makers since in sports games there will always exist extraordinary situations which are not adequately represented in the model.

At the beginning of Chapter 3, an overview of existing work in the field of sport-strategy optimization is given. This literature overview does not include purely statistical investigations. In sports science, there exists much literature that uses statistical methods to, e.g., compare playing characteristics (Koch and Tilp, 2009b). Some statistical investigations regarding beach volleyball or volleyball, which will be the investigated type of sports in this thesis, are presented in Chapter 4. However, the main literature overview presented in Chapter 3 focuses on work that tries to model the system dynamics explicitly. The mathematical tool that is used most of the time to reflect the system dynamics of a sports game is a Markov process. One notices when going through the presented examples that many works investigate a general principle (Turocy, 2008; Anbarci, Sun, and Ünver, 2015; Wright and Hirotsu, 2003), or the optimal decision rule is determined in dependence of some probabilities that are difficult to estimate a-priori a match. For example, Norman (1985) and Chan and Singal (2016) require the point-winning probabilities in tennis of each player as an input. If a particular pairing of teams or players has not occurred recently, it may be hard to estimate those required point-winning probabilities since they depend on both teams or players participating in the match.

In my opinion, there exists a research gap for methods that are applicable prior to a match in connection to a particular opponent. Alternatively, at least methods that include a procedure how the required input probabilities can be estimated prior to a match. This thesis tries to narrow or close this described gap.

The development of this thesis started with the idea to use Markov decision processes (MDPs) to handle sport-strategic decision problems. The choice of beach volleyball as the first considered sports game was quickly made since beach volleyball offers a good mixture of predefined structure and player interactions. Also, the decision question that should act as a benchmark question and be answered by the derived MDPs was quickly determined. Without any formalization, the benchmark question to be answered is: “Does *risky* play or a more *safer* play lead to a higher winning probability in a particular match against a certain opponent team?” This question was found to be an appropriate benchmark question since from a bird’s eye view there exists a trade-off between a higher probability for scoring but at the same time also a higher risk of failing. In chronological order according to the creation of the chapters, Chapter 4 that captures two MDPs for beach volleyball would have been the first chapter. A reader who is mainly interested in the application can start reading in Chapter 4. The theoretical background that is needed is linked at the appropriate places.

There always exist several opportunities to model the same question appropriately when modeling a decision problem. So, two models have been developed that model the same benchmark question. The models differed in their granularity, and it was soon apparent that each level of granularity had something to offer. So both models have been pushed to the most extreme levels of granularity such that the benchmark question was still appropriately modeled, but the advantages and disadvantages of the different granularity levels appear clearly. The two models are called set-SSO-MDP and rally-SSO-MDP and are both presented in Chapter 4 where their strengths and weaknesses are described. Especially, when trying to apply those models to a real match situation these strengths and weaknesses appear. The final match of the beach volleyball tournament at the Olympic games 2012 in London was used as a case study for the described benchmark question. Collecting appropriate input-data for both models contained some unexpected hurdles. Besides software tools that were needed to support the data collection process, also routines to check the integrity of the significant number of observations needed to be developed. Furthermore, general concepts, like an aggregation scheme, were developed and adjustments of the models were made to make them suitable for data collected from real matches.

After having developed first models for sport-strategy optimization problems, there arose two paths that were followed. The first direction was to identify similarities in the developed models to characterize a class of MDPs that might generally be suitable to model sports games. The second direction was to develop a method that combines the strengths of different model granularities.

The first path ended up with a definition of so-called *Sport-Strategy Optimization MDPs (SSO-MDPs)*, see Definition 3.2.2. Those MDPs are constructed such that strategies with the highest probability of winning the considered sports game are optimal strategies. Also, an important characteristic of SSO-MDPs was identified: Every action in an SSO-MDPs relates in some way to a physical action of a player and has therefore always a positive probability to fail. This observed characteristic is captured in an assumption on SSO-MDPs. From this assumption, a special structure can be followed for SSO-MDPs. By exploiting this structure, convergence results of solution algorithms, a special linear programming formulation, and a transformation algorithm could be developed. Those theoretical results related to SSO-MDPs are presented in Chapter 3. For completeness, the underlying basic definitions and theory

of general MDPs are presented in Chapter 2. Chapter 2 may be a good starting point for readers that search for a general introduction to MDPs. Those who are familiar with the general MDP-theory may skip that chapter and start reading in Chapter 3 on the theoretical properties of SSO-MDPs.

The second direction examined led to the so-called *Two-Scale Approach (2-MDP approach)*, which combines the advantages of two models with a different granularity in one procedure. The general description and motivation of this procedure are presented in Chapter 5. The coarser of the two models is called the *strategic-MDP (s-MDP)* whereas, the more detailed model is called the *gameplay-MDP (g-MDP)*. The combined models need to capture the same sport-strategic question and a particular relation called *s-g-implementation* between states and actions in both models needs to be specified. Afterward, in Section 5.2, the 2-MDP approach was implemented for the two models of different granularity developed for beach volleyball. It was evaluated for the benchmark question concerning the final match of the Olympic games. The results were compared to those derived from a single model. Besides that, the 2-MDP approach gives new opportunities for analyzing a sports game. Those opportunities are presented at the end of Chapter 5 and show the strengths of the new procedure.

The last chapter, Chapter 6, concludes this thesis with a summary and valuation of the results. An outlook to further developments is given.

Chapter 2

Theory of Markov Decision Problems

This chapter provides an overview on the theory of Markov decision problems (MDPs). It focuses on infinite-horizon MDPs under the total expected reward criterion as they will be the main tool for analyzing a sport-strategic question in the next chapter. Before starting with the basic definition of an MDP, a short overview of the most well-known results from the MDP theory is given. Further references on more detailed results are given in the individual sections in this chapter.

One of the first occurrences of MDPs in the literature are Bellman (1957a) and Bellman (1957b), where Bellman used dynamic programming to compute the optimal value function of an MDP and analyzed its asymptotic behavior. He proved the fundamental result that there exists an optimal stationary policy and developed the value iteration algorithm for solving MDPs.

Howard (1960) wrote a book on MDPs that includes a computational technique for solving Markov decision processes called policy iteration.

D'Epenoux (1963) discovered that MDPs can be formulated by linear programs and therefore be solved in (weakly) polynomial time by either the ellipsoid method or the interior-point algorithm (Karmarkar, 1984; Khachiyan, 1980).

Papadimitriou and Tsitsiklis (1987) showed that the complexity of MDPs (finite horizon, infinite horizon discounted, infinite horizon average cost) are P-complete. Furthermore, they showed that for MDPs with deterministic transition functions the problem is strongly polynomial. The algorithms solved the deterministic MDPs very fast in parallel.

A modification of the policy iteration called *simple policy iteration* is equivalent to the simplex method of Dantzig when applied to solve an MDP. In the simple policy iteration the policy is only updated in a single state whereas the classic policy iteration corresponds to a block pivoting simplex algorithm. The simplex method (Dantzig, 1963) is the most popular algorithm for solving linear programs and performs well in practice. However, it has been shown that Dantzig's pivoting rule can lead to an exponential number of iterations (Klee and Minty, 1972). The counterexamples for certain simplex pivoting rules can not be directly transferred to MDPs since the set of linear programs resulting from MDPs may not include the counterexamples (Littman, Dean, and Kaelbling, 1995). Indeed, Ye (2011) showed that the simplex method with the most-negative-reduced-cost pivoting rule of Dantzig are strongly polynomial-time algorithms for solving MDPs with a fixed discount rate. As the classical policy iteration outperforms simple policy iteration (Littman, Dean, and Kaelbling, 1995), the classic policy-iteration method is also a strongly polynomial-time algorithm for solving the MDPs with a fixed

discount rate.

Well-known textbooks on MDPs are Puterman (2005), Bertsekas (2005), and Bertsekas (2001).

2.1 Basic Definition of Markov Decision Problems (MDPs)

In this section the basic notation for Markov decision problems (MDPs) which will be use in the following of this work is defined. Most of the notation is according to Puterman (2005). A Markov decision problem is a Markov decision process together with an optimality criterion.

2.1.1 Markov Decision Processes

A Markov decision process is defined in the following way:

Definition 2.1.1 (Markov Decision Process).

A discrete-time Markov decision process is a collection of objects

$$(T, S, \mathcal{A}, p_t(\cdot|s, a), r_t(s, a))$$

with the following meaning:

- $T = \{1, 2, \dots, N - 1\}$, $N \leq \infty$ is the set of decision epochs.
- S is the set of possible system states.
- $\mathcal{A} = \cup_{s \in S} \mathcal{A}_s$ is the set of all actions, where \mathcal{A}_s defines the set of allowable actions in state s .
- $p_t(\cdot|s, a)$ is the transition probability distribution function in state $s \in S$ at time $t \in T$ under action $a \in \mathcal{A}_s$.
- $r_t(s, a)$ is the expected reward when choosing action $a \in \mathcal{A}_s$ in state $s \in S$ at decision epoch $t \in T$; $r_N(s)$ is the terminal reward of a finite horizon Markov decision process ($N < \infty$) when the process ends in state $s \in S$ at decision epoch N .

Throughout this thesis, only Markov decision processes with a finite state and finite action set are considered. From now on, it will be assumed that $|S| = n < \infty$ as well as $|\mathcal{A}| = m < \infty$.

The transition probability distribution function $p_t(\cdot|s, a)$ denotes by

$$p_t(j|s, a)$$

the probability that the system evolves in the next epoch to state j when it is at time t in state s and action a is chosen. Since $p_t(\cdot|s, a)$ defines a probability distribution over the set of states, it holds

$$\sum_{j \in S} p_t(j|s, a) = 1, \quad \forall t \in T, \quad \forall a \in \mathcal{A}_s.$$

The expected reward $r_t(s, a)$ of choosing action a in state s at time t can be calculated from the reward received for a transition from state s to j under action a , which is denoted by $r_t(s, a, j)$, in the following way:

$$r_t(s, a) = \sum_{j \in \mathcal{S}} r_t(s, a, j) p_t(j|s, a), \quad \forall s \in \mathcal{S}, \quad \forall t \in T.$$

We call a Markov decision process *stationary* if $p_t(j|s, a) = p(j|s, a)$, $\forall t \in T$ for the transition function as well as $r_t(s, a, j) = r(s, a, j)$, $\forall t \in T$ for the reward function holds.

A decision rule specifies how a decision maker in a Markov decision process selects an action. In general, a decision rule may depend on the complete history of the Markov decision process. However, in this thesis only Markov decision rules, which are decision rules that depend only on the current state of the system, are considered.

Definition 2.1.2 (Markovian Decision Rule).

A Markovian decision rule is a function

$$d_t : \mathcal{S} \rightarrow \mathcal{P}(A_s),$$

which specifies for each state a probability distribution over the set of available actions in that state. When using decision rule d_t and the system occupies state s at decision epoch t , an action a is chosen according to the specified probability distribution.

If the probability distribution $q_{d_t(s)}(\cdot) \in \mathcal{P}(A_s)$ is degenerated, i.e., for some $a \in A_s$ the probability of choosing that action in state s at time t is $q_{d_t(s)}(a) = 1$, the decision rule is called *deterministic*. For a randomized decision rule, the expected rewards satisfy

$$r_t(s, d_t(s)) = \sum_{a \in A_s} r_t(s, a) q_{d_t(s)}(a)$$

and the transition probabilities satisfy

$$p_t(j|s, d_t(s)) = \sum_{a \in A_s} p_t(j|s, a) q_{d_t(s)}(a).$$

Definition 2.1.3 (Markovian Policy).

A Markovian policy π is a sequence of Markovian decision rules, i.e.,

$$\pi = (d_1, d_2, \dots, d_{N-1})$$

where d_t for $t = 1, 2, \dots, N - 1$ is a Markovian decision rule.

A policy is *stationary* if $d_t = d$ for all $t \in T$. We abbreviate $\pi = (d, d, d, \dots)$ by d^∞ .

For MDPs, it can be shown that for any given history-dependent policy and starting state, a randomized Markov policy can be constructed that yields the same reward stream (Puterman, 2005, Thm.

5.5.1). D^{MR} is defined as the set of all randomized Markov decision rules, $D^{MD} \subseteq D^{MR}$ as the set of all deterministic Markov decision rules and analogously Π^{MR} and $\Pi^{MD} \subseteq \Pi^{MR}$ as the sets of randomized respectively deterministic Markov policies.

A Markov decision process together with a Markovian policy π and an initial distribution P_1 induces a stochastic Markov process. Consider a probability model (Ω, F, \mathbb{P}) where Ω denotes a sample space, F a σ -algebra over Ω and \mathbb{P} a probability measure on F . The sample space of the stochastic process induced by a Markov decision process is $\Omega := \{S \times \mathcal{A}\}^{N-1} \times S$ respectively $\Omega := \{S \times \mathcal{A}\}^\infty$ for an infinite horizon Markov decision process. For finite sets S and \mathcal{A} , the sample space Ω contains at most countably many elements and F can be chosen as the set of all subsets of Ω , which is denoted by 2^Ω .

The pairs $(S, 2^S)$ and $(\mathcal{A}, 2^{\mathcal{A}})$ are discrete measure spaces. The random variables X_t and Y_t on $(\Omega, 2^\Omega, \mathbb{P})$ are defined by

$$\begin{aligned} X_t : \Omega &\rightarrow S & Y_t : \Omega &\rightarrow \mathcal{A} \\ \omega &\mapsto X_t(\omega) = s_t & \omega &\mapsto Y_t(\omega) = a_t. \end{aligned}$$

Let $\pi = (d_1, d_2, \dots) \in \Pi^{MR}$ be a randomized Markov policy and P_1 an initial distribution of the system state. Then π induces a probability distribution \mathbb{P}^π on $(\Omega, 2^\Omega)$. A summary of the knowledge about the distribution of X_t and Y_t in an MDP under a policy π to determine \mathbb{P}^π is given in the following: The probability of the initial state is given through P_1 :

$$\mathbb{P}^\pi \{X_1 = s\} = P_1(s).$$

Since π is assumed to be a Markov policy, the probability of an action $a \in \mathcal{A}$ at time t is only depending on the state s_t and given by

$$\begin{aligned} &\mathbb{P}^\pi \{Y_t = a \mid X_1 = s_1, Y_1 = a_1, \dots, X_t = s_t\} \\ &= \mathbb{P}^\pi \{Y_t = a \mid X_t = s_t\} \\ &= q_{d_t(s_t)}(a). \end{aligned}$$

Due to the nature of a Markov decision process, the transition probabilities depend only on the last state and the chosen action. The probability of a transition from state s_t to state s_{t+1} under action a_t equals

$$\begin{aligned} &\mathbb{P}^\pi \{X_{t+1} = s_{t+1} \mid X_1 = s_1, Y_1 = a_1, \dots, X_t = s_t, Y_t = a_t\} \\ &= \mathbb{P}^\pi \{X_{t+1} = s_{t+1} \mid X_t = s_t, Y_t = a_t\} \\ &= p_t(s_{t+1} | s_t, a_t). \end{aligned}$$

This determines the probability of an event $\omega = (s_1, a_1, s_2, \dots, s_N) \in \Omega$ as

$$\begin{aligned} &\mathbb{P}^\pi \{(s_1, a_1, s_2, \dots, s_N)\} \\ &= P_1(s_1) \cdot q_{d_1(s_1)}(a_1) \cdot p_1(s_2 | s_1, a_1) \cdot \dots \cdot q_{d_{N-1}(s_{N-1})}(a_{N-1}) \cdot p_{N-1}(s_N | s_{N-1}, a_{N-1}). \end{aligned}$$

$\{X_t, t \in T\}$ and $\{Y_t, t \in T\}$ is called the induced stochastic process by the Markov decision process and the policy π . Moreover, $\{X_t, t \in T\}$ is a discrete time Markov chain since it satisfies the Markov property. For

$$\mathbb{P}^\pi \{(s_1, a_1, \dots, s_t)\} > 0,$$

the Markov property holds since

$$\begin{aligned}
& \mathbb{P}^\pi \{(a_t, s_{t+1}, \dots, s_n) \mid (s_1, a_1, \dots, s_t)\} \\
&= \frac{\mathbb{P}^\pi \{(s_1, a_1, \dots, s_n)\}}{\mathbb{P}^\pi \{(s_1, a_1, \dots, s_t)\}} \\
&= \frac{P_1(s_1) \cdot q_{d_1(s_1)}(a_1) \cdot \dots \cdot q_{d_t(s_t)}(a_t) \cdot \dots \cdot p_{n-1}(s_n \mid s_{n-1}, a_{n-1})}{P_1(s_1) \cdot q_{d_1(s_1)}(a_1) \cdot \dots \cdot p_{t-1}(s_t \mid s_{t-1}, a_{t-1})} \\
&= q_{d_t(s_t)}(a_t) \cdot \dots \cdot p_{n-1}(s_n \mid s_{n-1}, a_{n-1}) \\
&= \mathbb{P}^\pi \{(a_t, s_{t+1}, \dots, s_n) \mid s_t\}.
\end{aligned}$$

This shows that the evolution of the process depends only on the current state s_t and not on the full history (s_1, a_1, \dots, s_t) .

The stochastic process $\{(X_t, r_t(X_t, Y_t)), t \in T\}$ is called the Markov reward process. It is a sequence of states and a stream of expected rewards generated by the Markovian policy π .

For a real valued random variable W on the probability model $(\Omega, F, \mathbb{P}^\pi)$, the expected value of W is defined as

$$\mathbb{E}^\pi \{W\} := \sum_{\omega \in \Omega} W(\omega) \cdot \mathbb{P}^\pi \{\omega\} = \sum_{w \in \mathbb{R}} w \cdot \mathbb{P}^\pi \{\omega : W(\omega) = w\}.$$

2.1.2 Optimality Criteria

Popular optimality criteria for Markov decision processes are the *expected total reward criterion*, the *expected discounted reward criterion* and the *average reward criterion*. The expected total reward criterion is often used for finite as well as for infinite horizon Markov decision processes. The expected discounted and the average reward criterion are mainly used for infinite horizon Markov decision processes.

Let X_t be a random variable that represents the state of the system at time t and Y_t be a random variable that represents the selected action at time t . Assume that policy $\pi \in \Pi^{MR}$ is followed and the system starts in a fixed state s .

Definition 2.1.4 (Expected Total Reward Criterion).

In the setting of a finite horizon Markov decision process ($N < \infty$) the expected total reward of policy π is defined as

$$v_N^\pi(s) = \mathbb{E}_s^\pi \left\{ \sum_{t=1}^{N-1} r_t(X_t, Y_t) + r_N(X_N) \right\}.$$

In the setting of an infinite horizon Markov decision process, where no terminal reward exists, the expected total reward of policy π is defined as

$$v^\pi(s) = \lim_{N \rightarrow \infty} \mathbb{E}_s^\pi \left\{ \sum_{t=1}^N r_t(X_t, Y_t) \right\} = \lim_{N \rightarrow \infty} v_{N+1}^\pi(s). \quad (2.1)$$

If an initial distribution $P_1 \in \mathcal{P}(S)$ of the starting state is given, the expected total reward of a policy π is defined as

$$\sum_{s \in S} P_1(s) v_N^\pi(s) \text{ resp. } \sum_{s \in S} P_1(s) v^\pi(s).$$

The same applies to the following reward criteria.

Definition 2.1.5 (Expected Discounted Reward Criterion).

The expected discounted reward of policy π is defined as

$$v_\lambda^\pi(s) = \lim_{N \rightarrow \infty} \mathbb{E}_s^\pi \left\{ \sum_{t=1}^N \lambda^{t-1} r_t(X_t, Y_t) \right\} \quad (2.2)$$

for $0 \leq \lambda < 1$.

Definition 2.1.6 (Average Reward Criterion).

The average reward of policy π is defined as

$$g^\pi(s) = \lim_{N \rightarrow \infty} \frac{1}{N} \mathbb{E}_s^\pi \left\{ \sum_{t=1}^N r_t(X_t, Y_t) \right\} = \lim_{N \rightarrow \infty} \frac{1}{N} v_{N+1}^\pi(s). \quad (2.3)$$

For general infinite horizon Markov decision processes, the limits in Equations 2.1, 2.2 and 2.3 need not exist. Some criteria for the expected discounted and the average reward criterion that ensure that the limits exist are summarized in the following:

- The limit in the definition of the expected discounted reward criterion (2.2) exists when the expected reward function is bounded by some finite constant (Puterman, 2005, Sec. 5.1).
- The limit in the definition of the average reward criterion (2.3) exists for Markov decision processes with a finite state space (Puterman, 2005, Sec. 8.1.2).

In Subsection 2.3.1, different classes of Markov decision processes under the expected total reward criterion are investigated. The Markov decision processes of the considered classes fulfill characteristics which ensure the existence of $v^\pi(s)$.

When the limit in Equation 2.1 exists and the interchange of the limit with the expectation is valid, it is possible to write

$$v^\pi(s) = \mathbb{E}_s^\pi \left\{ \sum_{t=1}^{\infty} r_t(X_t, Y_t) \right\}.$$

In an MDP, a policy with the largest value function is sought. In the following Definition 2.1.7, an optimal policy under the introduced optimality criteria is characterized. For the expected total reward criterion, the finite horizon case is stated in brackets.

Definition 2.1.7 (Optimal Policy).

A policy π^* is *total reward optimal* if π^* satisfies

$$v^{\pi^*}(s) \geq v^\pi(s) \left[v_N^{\pi^*}(s) \geq v_N^\pi(s) \right] \quad \forall s \in \mathcal{S}, \forall \pi \in \Pi^{MR}.$$

A policy π^* is *discount optimal* if for fixed λ , $0 \leq \lambda < 1$,

$$v_\lambda^{\pi^*}(s) \geq v_\lambda^\pi(s) \quad \forall s \in \mathcal{S}, \forall \pi \in \Pi^{MR}.$$

A policy $\pi^* \in \Pi$ is *average optimal* if π^* satisfies

$$g^{\pi^*}(s) \geq g^\pi(s) \quad \forall s \in \mathcal{S}, \forall \pi \in \Pi^{MR}.$$

Definition 2.1.8 (Value of an MDP).

The *value* of an MDP is defined by

$$v^*(s) = \sup_{\pi \in \Pi^{MR}} v^\pi(s) \left[v_N^*(s) = \sup_{\pi \in \Pi} v_N^\pi(s) \right], \quad s \in \mathcal{S}.$$

The value of an MDP under the discounted or average reward criterion is defined analogously.

An optimal policy $\pi^* \in \Pi^{MR}$ exists when

$$v^{\pi^*}(s) = v^*(s) \quad \forall s \in \mathcal{S}.$$

Observe, that an optimal policy achieves the value of the MDP for each initial state $s \in \mathcal{S}$.

2.2 Finite Horizon MDPs

In this subsection a finite horizon MDP ($N < \infty$) under the expected total reward criterion is assumed. Dynamic programming (or backward induction) is the fundamental recursion for evaluating policies. Together with the optimality equations, it is an efficient method for determining an optimal policy in finite-horizon MDPs.

2.2.1 Policy Evaluation

First, Algorithm 1 describes how a deterministic policy $\pi = (d_1, d_2, \dots, d_{N-1}) \in \Pi^{MD}$ can be evaluated by dynamic programming.

Algorithm 1: Finite-Horizon Policy Evaluation

Data: Finite-Horizon MDP $(T, S, \mathcal{A}, p_t(\cdot|s, a), r_t(s, a)), N < \infty, \pi \in \Pi^{MD}$
 Result: $u_1^\pi(s) = v_N^\pi(s)$

- 1 $t \leftarrow N$;
- 2 $u_N^\pi(s_N) \leftarrow r_N(s_N)$ for all $s_N \in S$;
- 3 while $t > 1$ do
- 4 $t \leftarrow t - 1$;
- 5 foreach $s_t \in S$ do
- 6 Compute $u_t^\pi(s_t) = r_t(s_t, d_t(s_t)) + \sum_{j \in S} p_t(j|s_t, d_t(s_t))u_{t+1}^\pi(j)$;
- 7 end
- 8 end

As a result, the policy evaluation algorithm computes the expected total reward of a fixed policy π . The function $u_t^\pi(s_t)$ equals the expected total reward obtained by using policy π from decision epoch t onwards starting in state s_t :

$$u_t^\pi(s_t) = \mathbb{E}_{s_t}^\pi \left\{ \sum_{n=t}^{N-1} r_t(X_n, d_n(X_n)) + r_N(X_N) \right\},$$

where X_n is random variable representing the state of the system at time n . This relation can be verified by backward induction (Puterman, 2005, Thm. 4.2.1). As a consequence $u_1^\pi(s) = v_N^\pi(s)$ for all $s \in S$ which is the expected total reward of policy π .

The policy evaluation algorithm can be generalized to randomized decision rules by replacing the equation in line 6 by

$$u_t^\pi(s_t) = \sum_{a \in \mathcal{A}_{s_t}} q_{d_t(s_t)}(a) \left\{ r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a)u_{t+1}^\pi(j) \right\}.$$

The policy evaluation algorithm calculates the expected total reward of a policy by evaluating $N - 1$ times a one-period calculation. Assume, there are n possible states for s_t at each time step t . Then the policy evaluation algorithm evaluates $(N - 1) \cdot n$ equations where the sum in each equation is calculated over n realizations of the next state.

An enumeration over all realizations would require evaluating the expectation over the joint probability distribution of $n^{(N-1)}$ realizations under policy π . The reduction of an $(N - 1)$ -period problem to $(N - 1)$ many 1-period problems is the key idea of dynamic programming. It is also used for finding an optimal policy.

2.2.2 Optimality Equations and Backward Induction

The optimality equations (or Bellmann equations) are the basis for determining optimal policies by dynamic programming. They are given by

$$u_t(s_t) = \max_{a \in \mathcal{A}_{s_t}} \left\{ r_t(s_t, a) + \sum_{j \in \mathcal{S}} p_t(j|s_t, a) u_{t+1}^\pi(j) \right\} \quad (2.4)$$

$$u_N(s_N) = r_N(s_N). \quad (2.5)$$

Let

$$u_t^*(s_t) = \max_{\pi \in \Pi^{MR}} u_t^\pi(s_t)$$

be the optimal value function from period t on. Let $u_t(\cdot)$ be a solution of equation 2.4 and 2.5. Then $u_t(s_t) = u_t^*(s_t)$ for all $s_t \in \mathcal{S}$, $t = 1, \dots, N$ and $u_1(s_1) = v_N^*(s_1)$ (Puterman, 2005, Thm. 4.3.2) which means that a solution of the optimality equations gives the optimal reward from period t onwards and the value of the MDP. The principle of optimality described by the optimality equations can be verbally stated by:

“An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.” (Bellman, 1957b)

However, there exist optimality criteria for which the optimality equations do not hold.

Any policy that uses a decision rule

$$d_t(s_t) \in \arg \max_{a \in \mathcal{A}_{s_t}} \left\{ r_t(s_t, a) + \sum_{j \in \mathcal{S}} p_t(j|s_t, a) u_{t+1}^*(j) \right\}$$

is an optimal policy. For MDPs with a finite state set and a finite action set, there always exists a deterministic Markovian policy which is optimal (Puterman, 2005, Thm. 4.3.3, Prop. 4.4.3). The mentioned theorem is helpful since it allows to restrict to deterministic Markov policies, which are easier to implement and evaluate than randomized Markov policies.

The backward induction algorithm, Algorithm 2, solves the optimality equations and saves in each iteration the actions at which the maximum in equation 2.4 is attained. Any policy that selects only actions from the optimal action sets $\mathcal{A}_{s_t, t}^*$ is an optimal policy. So, with the backward induction algorithm the value of the MDP as well as optimal policies can be computed.

For $|\mathcal{S}| = n$ and $|\mathcal{A}| = m$ the backward induction algorithm requires $(N - 1) \cdot n$ iterations. Where in each iteration a maximum over m actions is computed. And each computation requires n multiplications for calculating the 1-period expectation. In total $(N - 1) \cdot m \cdot n^2$ multiplications are needed to compute an optimal policy. In contrast, there are $m^{n(N-1)}$ deterministic Markovian policies. For directly evaluating one of these policies, one would need $(N - 1) \cdot n^2$ multiplications. So, the

backward induction algorithm decreases the number of computations significantly.

Algorithm 2: Backward Induction Algorithm

Data: Finite-Horizon MDP $(T, S, \mathcal{A}, p_t(\cdot|s, a), r_t(s, a))$, $N < \infty$
 Result: $u_1^*(s) = v_N^*(s)$

- 1 $t \leftarrow N$;
- 2 $u_N^*(s_N) \leftarrow r_N(s_N)$ for all $s_N \in S$;
- 3 while $t > 1$ do
- 4 $t \leftarrow t - 1$;
- 5 foreach $s_t \in S$ do
- 6 Compute $u_t^*(s_t) = \max_{a \in \mathcal{A}_{s_t}} \left\{ r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(j) \right\}$;
- 7 Set $\mathcal{A}_{s_t, t}^* = \arg \max_{a \in \mathcal{A}_{s_t}} \left\{ r_t(s_t, a) + \sum_{j \in S} p_t(j|s_t, a) u_{t+1}^*(j) \right\}$;
- 8 end
- 9 end

2.3 Infinite-Horizon Expected Total Reward MDPs

In this sections on infinite-horizon MDPs focuses on infinite-horizon expected total reward MDPs with finite sets of states and actions. The infinite horizon will be omitted in the MDP description. Furthermore, an MDP with *stationary* problem data is assumed. From a finite number of states and actions, it follows that the reward function takes only finitely many values. Each value of the reward function is assumed to be finite and can therefore be bounded by some constant. Assumption 2.3.1 summarizes the assumptions made in this section.

Assumption 2.3.1:

The assumptions for this section on infinite-horizon MDPs under the expected total reward criterion are

- The horizon $N = \infty$.
- The decision process has stationary problem data $p(\cdot|s, a)$ and $r(s, a)$.
- The set of states and the set of actions are finite. Let $|S| = n$ be the number of states and $|\mathcal{A}| = m$ be the number of actions.
- The reward function is bounded by a constant M , i. e.,

$$|r(s, a, s')| \leq M, \quad \forall s, s' \in S, a \in \mathcal{A}_s.$$

Let $r^+(s, a) = \max\{r(s, a), 0\}$ and $r^-(s, a) = \max\{-r(s, a), 0\}$ and define

$$v_{\pm}^{\pi}(s) = \lim_{N \rightarrow \infty} \mathbb{E}_s^{\pi} \left\{ \sum_{t=1}^N r^{\pm}(X_t, Y_t) \right\}$$

and

$$v_-^\pi(s) = \lim_{N \rightarrow \infty} \mathbb{E}_s^\pi \left\{ \sum_{t=1}^N r^-(X_t, Y_t) \right\}.$$

For $v_+^\pi(s)$ or $v_-^\pi(s)$ finite, the limit $v^\pi(s) = \lim_{N \rightarrow \infty} v_{N+1}^\pi(s)$ is well-defined and

$$v^\pi(s) = v_+^\pi(s) - v_-^\pi(s).$$

For this chapter and the rest of this thesis, we will only consider MDPs where the following assumption is satisfied such that the expected total reward is well-defined for each policy π :

Assumption 2.3.2 (Well defined expected total reward):

For all $\pi \in \Pi^{MR}$ and $s \in S$, $v_+^\pi(s)$ or $v_-^\pi(s)$ is finite.

This section starts by presenting different classes of infinite-horizon MDPs. Afterwards, a vector notation and the optimality equations for infinite-horizon MDPs are introduced. Furthermore, this section presents a linear programming formulation for infinite-horizon MDPs and the algorithms value iteration and policy iteration that can be used for determine an optimal policy of an infinite-horizon MDP.

2.3.1 Classification

There exist some investigated classes of infinite-horizon expected total reward MDPs, which are introduced in this subsection. The presented MDP classes come from different authors and are adapted to the reward based notation of this thesis.

Positive bounded and negative models

Puterman (2005) focuses on two classes called *positive bounded* models and *negative* models. The idea behind the two classes of MDPs is to guarantee that the sum of positive or the sum of negative rewards is finite.

Definition 2.3.3 (Positive Bounded Model).

An MDP $(S, \mathcal{A}, p(\cdot|s, a), r(s, a))$ is a *positive bounded model (POSB)* if the following holds:

- $v_+^\pi(s) < \infty$ for all $s \in S$ and $\pi \in \Pi^{MR}$, and
 - For each $s \in S$ there exists at least one $a \in \mathcal{A}_s$ with $r(s, a) \geq 0$.
-

Definition 2.3.4 (Negative Model).

An MDP $(S, \mathcal{A}, p(\cdot|s, a), r(s, a))$ is a *negative model (NEG)* if the following holds:

- $v_+^\pi(s) = 0$ for all $s \in S$ and $\pi \in \Pi^{MR}$, and
 - There exists a policy $\pi \in \Pi^{MR}$ with $v^\pi(s) > -\infty$ for all $s \in S$.
-

Maximizing negative expected rewards is equivalent to minimizing the expected costs. So, negative models can be formulated for problems of minimizing non-negative costs.

Stochastic Shortest Path Problems

Bertsekas (2001) defines a stochastic shortest path problem. Usually, stochastic shortest path problems are defined in a minimizing expected cost environment. However, for consistency, the reward function is defined as the negative cost function and the notation is adapted to the one used for maximization problems:

Definition 2.3.5 (Stochastic Shortest Path Problem).

An MDP

$$(S, \mathcal{A}, p(\cdot|s, a), r(s, a))$$

together with a set of goal states $G \subseteq S$ is a *stochastic shortest path (SSP)* problem

$$(S, \mathcal{A}, p(\cdot|s, a), r(s, a), G)$$

if it satisfies

- $p(g|g, a) = 1$ for all $g \in G, a \in \mathcal{A}_g$
 - $r(g, a) = 0$ for all $g \in G, a \in \mathcal{A}_g$
-

The first property ensures that each goal state $g \in G$ is absorbing, i. e., once the process has entered g , it will stay in g . From the second property, it follows that no rewards are accumulated when staying in g .

Bertsekas makes two more assumptions on the considered SSP MDPs:

Assumption 2.3.6:

Assumptions on SSP MDPs considered by Bertsekas are

1. *There exists at least one complete proper policy, i. e., a stationary policy d^∞ with*

$$\mathbb{P}^{d^\infty} \{X_n \notin G \mid X_1 = s\} < 1, \quad \forall s \in S$$

where $n = |S|$ and \mathbb{P}^{d^∞} means that decision rule d is followed at each decision epoch.

2. *For every improper policy d^∞ , i. e., for every policy that is not proper, the reward $v^{d^\infty}(s)$ is $-\infty$ for at least one state $s \in S$.*

Note that in the terminology of this thesis an SSP MDP is just an MDP with goal states without any further assumptions. An SSP MDP that satisfy Assumption 2.3.6 is called a *Bertsekas-SSP* MDP:

Definition 2.3.7 (Bertsekas-SSP MDP).

An SSP MDP that satisfies Assumptions 2.3.6 is a *Bertsekas-SSP* MDP.

The first condition of Assumption 2.3.6 assures that there exists a policy such that each state is connected to a goal state by a path with a positive probability. This condition implies that under such a complete proper policy a goal state is reached with probability 1 from every state s . This can be seen as follows: Define

$$\varrho_d := \max_{s \in \mathcal{S}} \mathbb{P}^{d^\infty} \{X_n \notin G \mid X_1 = s\},$$

then

$$\mathbb{P}^{d^\infty} \{X_k \notin G \mid X_1 = s\} \leq \varrho_d^{\lfloor \frac{k}{n} \rfloor}, \forall s \in \mathcal{S}. \quad (2.6)$$

For complete proper policies, $\varrho_d < 1$ holds. So, the probability of not reaching a goal state after k steps converges to zero as k goes to infinity.

For a bounded expected reward function, the value of a complete proper policy exists and is finite since the expected reward in the k -th period is bounded by

$$\varrho_d^{\lfloor \frac{k}{n} \rfloor} \max_{i=1, \dots, n} |r(s, d(s))| \leq \varrho_d^{\lfloor \frac{k}{n} \rfloor} \mathcal{M}.$$

For complete proper policies, the reward structure is similar to a Markov decision process under the discounted reward criterion. The difference is that the discount factor is not fixed. But, in stage $k \cdot n$ the discount factor is less or equal ϱ_d^k .

If there exists a dead end \bar{s} , i. e., a state that is not connected by a path to a goal state, the assumption of the existence of a complete proper policy can not be satisfied. Starting in this dead end would lead to $\mathbb{P}^{d^\infty} \{X_k \notin G \mid X_1 = \bar{s}\} = 1$ for all $k \in \mathbb{N}$ and all decision rules $d \in D^{MR}$.

In Bertsekas (2001), a policy that satisfies the first assumption is named a proper policy. However, some of the following MDP classes require *proper* policies that are policies which reach a goal state with a probability of 1 but only from a specified starting state s_1 . In those SSP MDPs, there can exist dead ends which are not connected to s_1 under a proper policy. So, for a better distinction, the policies that are proper in every state of the state space are called *complete proper* policies.

The second assumption is satisfied if each cycle that does not contain a goal state has negative expected rewards. Sometimes, the stronger assumption that the expected reward function is strictly negative except for transitions from the absorbing states is used. This assumption has the advantage that it can be easier verified.

Generalized Stochastic Shortest Path Problems

Kolobov et al. (2011) introduce a new class of MDPs, called *generalized stochastic shortest path (GSSP)* problem that generalizes the class of Bertsekas-SSP problems:

Definition 2.3.8 (Generalized Stochastic Shortest Path Problem).

An SSP MDP extended by a starting state $s_1 \in S$ is a tuple

$$(S, \mathcal{A}, p(\cdot|s, a), r(s, a), G, s_1).$$

It is a *generalized stochastic shortest path (GSSP)* problem if the following assumptions hold

1. There exists at least one *proper* policy rooted at s_1 , i. e., a stationary policy d^∞ that reaches a goal state with probability 1 starting from s_1 .
2. The expected sum of nonnegative rewards of any policy is bounded from above:

$$v_+^\pi(s) = \lim_{N \rightarrow \infty} \mathbb{E}_s^\pi \left\{ \sum_{t=1}^N r^+(X_t, Y_t) \right\} < \infty,$$

for every state s reachable from s_1 .

One difference between a Bertsekas-SSP MDP and a GSSP MDP is that the GSSP MDP only requires a proper policy in the known starting state while the Bertsekas-SSP MDP requires the existence of a *complete proper policy*. In a GSSP MDP, there could exist states $s \in S$ for which no proper policy exists. So, there may exist dead ends, but they may not be reachable under a proper policy starting from s_1 . In terms of Assumption 2.3.6, the first condition can be written as: There exists a policy d^∞ such that

$$\mathbb{P}\{X_n \notin G \mid X_1 = s, d\} < 1 \quad \forall s \text{ reachable from } s_1 \text{ by following } d,$$

and n is the number of reachable states from s_1 by following d .

The second assumptions of a GSSP MDP ensures that the value of any policy is well-defined since the value $v^\pi(s)$ of a policy exists whenever $v_+^\pi(s)$ or $v_-^\pi(s)$ is finite (Puterman, 2005, p. 279).

Furthermore, the reward maximizing policy does not need to be proper in a GSSP MDP. In contrast, the second assumption of 2.3.6 of Bertsekas on SSP MDPs guarantees that the reward maximizing policy in a Bertsekas-SSP MDP is proper. In a GSSP MDP, there could exist a 0-reward cycle. If additionally going to a goal state requires incurring a negative reward, a reward maximizing policy would stay in the 0-reward cycle and not go for a goal state. As a consequence Kolobov et al. (2011) formulates the objective of a GSSP MDP as finding the reward maximizing policy over all proper policies:

Definition 2.3.9 (Optimal Policy of GSSP MDP).

A policy π^* is an optimal policy in a GSSP MDP if it satisfies

$$v^{\pi^*}(s_1) \geq v^\pi(s_1) \quad \forall \text{ proper } \pi \in \Pi^{MR}.$$

Accordingly, the value of a GSSP MDP is defined as:

Definition 2.3.10 (Value of the GSSP MDP).

We define the *value* of the GSSP MDP by

$$v^*(s_1) = \sup_{\text{proper } \pi \in \Pi^{MR}} v^\pi(s_1).$$

Note that $\sup \emptyset = -\infty$. So, if no proper policy in state s_1 exists, then $v^*(s_1) = -\infty$.

SSPADE and SSPUDE

Kolobov, Mausam, and Weld introduce in their paper (Kolobov, Mausam, and Weld, 2012) two further MDP classes that extend the class of Bertsekas-SSP MDPs by dead ends. To be consistent with the previous sections, the definition is adapted to a reward based setting. The first class is named *stochastic shortest path problem with avoidable dead ends (SSPADE MDP)*:

Definition 2.3.11 (Shortest Path Problem with avoidable dead ends).

A *stochastic shortest path problem with avoidable dead ends (SSPADE MDP)* is an SSP MDP with a starting state $s_1 \in \mathcal{S}$

$$(\mathcal{S}, \mathcal{A}, p(\cdot|s, a), r(s, a), G, s_1),$$

where the following assumptions hold

1. There exists at least one *proper* policy rooted at s_1 .
2. Every *improper* policy must have a value of $-\infty$ in at least one state reachable from s_1 under that improper policy.

The SSPADE MDP definition relaxes the first condition for Bertsekas-SSP MDPs by not requiring a complete proper policy. However, the SSPADE MDP class is only a subclass of the GSSP MDP class since the second condition excludes zero-reward dead ends reachable from s_1 .

The second class of MDPs introduced in (Kolobov, Mausam, and Weld, 2012) is named *stochastic shortest path problem with unavoidable dead ends (SSPUDE MDP)*. This class contains SSP MDPs where the probability of running into a dead end starting at s_1 is positive for all policies. An SSPUDE MDP is a Bertsekas-SSP MDP where no proper policy at state s_1 exists, which is called a Bertsekas-SSP MDP that is *improper* in s_1 .

Definition 2.3.12 (Shortest Path Problem with unavoidable dead ends).

A *stochastic shortest path problem with unavoidable dead ends (SSPUDE MDP)* is an SSP MDP that is *improper* at s_1 extended by a penalty:

$$(\mathcal{S}, \mathcal{A}, p(\cdot|s, a), r(s, a), G, P, s_1).$$

The penalty $P \in \mathbb{R}^- \cup \{-\infty\}$ is incurred if an agent decides to abort the process in a non-goal state. Further, the second Bertsekas-SSP MDP definition must hold:

- Every improper policy π has a value $v^\pi(s)$ of $-\infty$ in every state $s \in \mathcal{S}$ where π is improper.

If $P > -\infty$ the MDP is called a *finite stochastic shortest path problem with unavoidable dead ends (fSSPUDE MDP)*. If $P = -\infty$ the MDP is called a *infinite stochastic shortest path problem with unavoidable dead ends (iSSPUDE MDP)*. In an fSSPUDE MDP, the price of ending in a dead end can be compensated while in an iSSPUDE MDP a dead end is truly irrecoverable.

Since in fSSPUDE MDPs there can exist non-dead-end state that yield a smaller reward than dead-ends, Kolobov, Mausam, and Weld define a modified value function for fSSPUDE MDPs as follows:

$$v_F^\pi(s) := \max \{P, v^\pi(s)\}.$$

This modified value function is always finite and can be interpreted as a lower bound on any state's reward. If a state has an expected reward of less or equal P the process aborts. An optimal policy in an fSSPUDE MDP is a policy π^* satisfying

$$v_F^{\pi^*}(s) = \max_{\pi \in \Pi^{MR}} v_F^\pi(s) \quad \forall s \in \mathcal{S}.$$

For iSSPUDE MDPs Kolobov, Mausam, and Weld define a two-ordered criteria by defining the reward of a state as an ordered pair

$$v_I^\pi(s) := (\mathbb{P}_G^\pi(s), [v^\pi | \mathbb{P}_G^\pi](s)),$$

where $\mathbb{P}_G^\pi(s)$ is called a goal-probability function that gives the probability of reaching the goal from state s under policy π :

$$\mathbb{P}_G^\pi(s) := \sum_{t=1}^{\infty} \mathbb{P}^\pi \{X_t = g \in G, X_{t'} = s' \notin G, \forall 1 \leq t' < t, X_1 = s\}.$$

The function $[v^\pi | \mathbb{P}_G^\pi]$ is the expected total reward restricted to states for which $\mathbb{P}_G^\pi(s) > 0$ holds. Kolobov, Mausam, and Weld define a random variable \tilde{X}_t^π that denotes a distribution over states s' with $\mathbb{P}_G^\pi(s') > 0$. Having that, $[v^\pi | \mathbb{P}_G^\pi]$ can be defined as

$$[v^\pi | \mathbb{P}_G^\pi](s) := \mathbb{E}_s^\pi \left\{ \sum_{t=0}^{\infty} r(\tilde{X}_t^\pi, Y_t) \right\}$$

This two ordered criterion is used as follows: A policy π' is preferable to π at s , which is written as $\pi(s) < \pi'(s)$, whenever $v_I^\pi(s) < v_I^{\pi'}(s)$. $v_I^\pi(s) < v_I^{\pi'}(s)$ holds whenever

$$\mathbb{P}_G^\pi(s) < \mathbb{P}_G^{\pi'}(s) \vee \left(\mathbb{P}_G^\pi(s) = \mathbb{P}_G^{\pi'}(s) \wedge [v^\pi | \mathbb{P}_G^\pi](s) < [v^{\pi'} | \mathbb{P}_G^{\pi'}](s) \right).$$

Hence, a policy is an optimal policy π^* of an iSSPUDE MDP if it satisfies

$$\pi^*(s) \in \arg \max_{< \pi} v_I^\pi(s) \quad \forall s \in \mathcal{S}.$$

Kolobov, Mausam, and Weld (2012) show that for big enough dead-end penalty costs the optimal policies of fSSPUDE and iSSPUDE are identical. Further, they show in Kolobov, Mausam, and Weld (2012) that every fSSPUDE MDP can be converted in a Bertsekas-SSP MDP with the same set of optimal policies.

MAXPROB

MAXPROB problems can be derived from GSSP MDPs that do not have any proper policy at state s_1 . If no proper policy at state s_1 exists, the value of the GSSP MDP is $-\infty$ since the supremum over an empty set of proper policies is defined as $-\infty$. To distinguish between the improper policies, that all do not reach a goal state with certainty, one tries to find the policy that has the highest probability of reaching a goal state. A MAXPROB problem can be derived from a GSSP MDP by assigning a reward of 0 to all transitions to non goal states and a reward of 1 to transitions to a goal state. Kolobov and Mausam (2012) define a MAXPROB problem in the following way:

Definition 2.3.13 (MAXPROB).

A MAXPROB MDP is an SSP MDP

$$(S, \mathcal{A}, p(\cdot|s, a), r(s, a), G)$$

where the reward function obeys two conditions:

1. $r(s, a, s') = 0, \forall s, s' \notin G, a \in \mathcal{A}_s$
2. $r(s, a, g) = 1, \forall s \notin G, g \in G, a \in \mathcal{A}_s$

The expected total reward $v^\pi(s)$ in a MAXPROB MDP equals the probability of reaching a goal state when policy π is used and the process starts in s . An optimal policy π^* is a policy that maximizes the probability of reaching a goal state, i.e.,

$$\pi^* \in \arg \max_{\pi \in \Pi^{MR}} \mathbb{P}_G^\pi(s) \quad \forall s \in S.$$

Hence, the optimal value function $v^*(s)$ of a MAXPROB MDP is 1 for all states where a proper policy exists and 0 for dead ends.

Summary

This subsection summarizes the hierarchy of the presented infinite-horizon MDP classes. Some relations between the MDP classes are already investigated in (Kolobov et al., 2011). The remaining relations, where no author is mentioned, are added by me. Remember that in this work an SSP MDP is defined as an infinite-horizon MDP with goal states without any further assumptions. In some literature, the term SSP MDP corresponds to our Bertsekas-SSP MDP definition. Since GSSP MDPs, SSPADE MDPs, Bertsekas-SSP MDPs, MAXPROB MDPs and SSPUDE MDPs are all infinite horizon MDPs with goal states, it is clear that they form subsets of the set of general SSP MDPs. POSB MDPs and NEG MDPs are not goal oriented, however it is possible to show that they can be converted into goal oriented SSP MDPs that actually meet the GSSP MDP assumptions. In the following, the relation between those MDPs classes will be proven and summarized.

Theorem 2.3.14:
 $\text{SSPUDE} \cap \text{GSSP} = \emptyset$

PROOF. An SSPUDE MDP is by definition 2.3.12 an SSP MDP that is improper at state s_1 , i.e., there exists no proper policy for the starting state s_1 . A GSSP MDP requires at least one proper policy rooted at s_1 . Therefore, there can not exist an SSP MDP that belongs to the class of SSPUDE MDPs and GSSP MDPs at the same time. ■

Theorem 2.3.15:
 $\text{Bertsekas-SSP} \subset \text{SSPADE} \subset \text{GSSP} \subset \text{SSP}$

PROOF. Since Bertsekas-SSP, SSPADE as well as GSSP problems impose additional conditions on the basic class of SSP problems, it is clear that they all form subsets of the basic SSP problems.

GSSP problems form a strict subset of SSP problems because they require the existence of a proper policy for a specified starting state s_1 . In contrast, SSP problems contain instances where there does not exist a proper policy from any state.

First it is shown that an SSPADE MDP is also a GSSP MDP. Both problems require a proper policy rooted in s_1 . So, an SSPADE MDP obviously satisfies the first condition of a GSSP problem. All proper policies terminate with probability 1 in a goal state. It was already remarked that the value of a proper policy is finite. Since all improper policies in an SSPADE MDP have a value of $-\infty$ in at least one state reachable from s_1 , the nonnegative reward $v_+^\pi(s)$ must be finite after Assumption 2.3.2 which states that either $v^\pi(s)_+$ or $v^\pi(s)_-$ has to be finite for each policy π . So in conclusion, all policies satisfy $v_+^\pi(s) < \infty$ for all s reachable from s_1 and the second condition of the definition of GSSP problems is satisfied.¹ The set of GSSP problems may contain problems that are no SSPADE MDPs: For example, a GSSP problem may contain a dead end with 0-reward reachable from s_1 such that there exists an improper policy rooted in s_1 that has a finite reward in every state reachable from s_1 .

Every Bertsekas-SSP MDP is an SSPADE MDP since in a Bertsekas-SSP MDP every improper policy has a reward of $-\infty$ in at least on state s . So, an improper policy rooted in s_1 must have a value of $-\infty$ in at least on state s reachable from s_1 . The set of Bertsekas-SSP problems is a strict subset of SSPADE MDPs since an SSPADE MDP requires only a proper policy rooted in the starting state s_1 while a Bertsekas-SSP MDP requires a proper policy for every state s in the state space. So, in an SSPADE MDP, there may exist states not reachable from s_1 under a proper policy for which no proper policy exist. ■

¹The relation SSPADE \subset GSSP can also be found in Theorem 2 of Kolobov and Mausam (2012). The proof is only sketched and a reference to Kolobov et al. (2011) is given. However in Kolobov et al. (2011), I can not find this proof.

Kolobov et al. (2011) investigate the relation of the non-goal-oriented MDP classes *NEG* and *POSB*:

Theorem 2.3.16 (Kolobov et al. (2011)):
 $NEG \subset GSSP$

Theorem 2.3.17 (Kolobov et al. (2011)):
 $POSB \subset GSSP$

The proofs of $NEG \subset GSSP$ and $POSB \subset GSSP$ are based on constructing a GSSP MDP from an original NEG or POSB MDP with identical optimal solutions.

The idea for proving the relation $POSB \subset GSSP$ is the following: Consider the reachability graph of a POSB MDP. It can be shown that in the directed reachability graph, there must exist at least one strongly connected component. That strongly connected component has no outgoing edge and all internal actions have an expected total reward of 0. So, the set of states in these strongly connected components can be defined as goal states of the derived GSSP MDP.

For NEG MDPs, it can be shown that the second assumption equals to requiring the existence of a proper policy. Analogously to POSB MDPs, NEG MDPs can be converted into GSSP MDPs.

Theorem 2.3.18 (Kolobov et al. (2011)):
 $MAXPROB \subset POSB$

PROOF. Every MAXPROB MDP is a POSB MDP: Since there exist only 0 and 1-reward actions, the first condition of POSB, that there must exist an action with a non-negative expected reward in every state, is obviously satisfied. Also $v_{\pi}^{\mathbb{E}}(s)$ is finite since a strictly positive reward is only distributed for a transition to a goal state. Since a goal state is an absorbing state, no further strictly positive rewards can be accumulated. ■

There is one important remark to be made on the last theorems. From the relation $MAXPROB \subset POSB$ and $POSB \subset GSSP$, one can conclude that MAXPROB MDPs are a subset of GSSP. However, in contrast to the POSB MDP definition a MAXPROB MDP is goal-oriented, which means that a set G of goal-states is explicitly specified. In the proof of $POSB \subset GSSP$, a set of goal states is constructed according to some properties. So, when considering a MAXPROB MDP as a GSSP, the set of goal states may have to be modified according to the proof of Theorem 2.3.17.

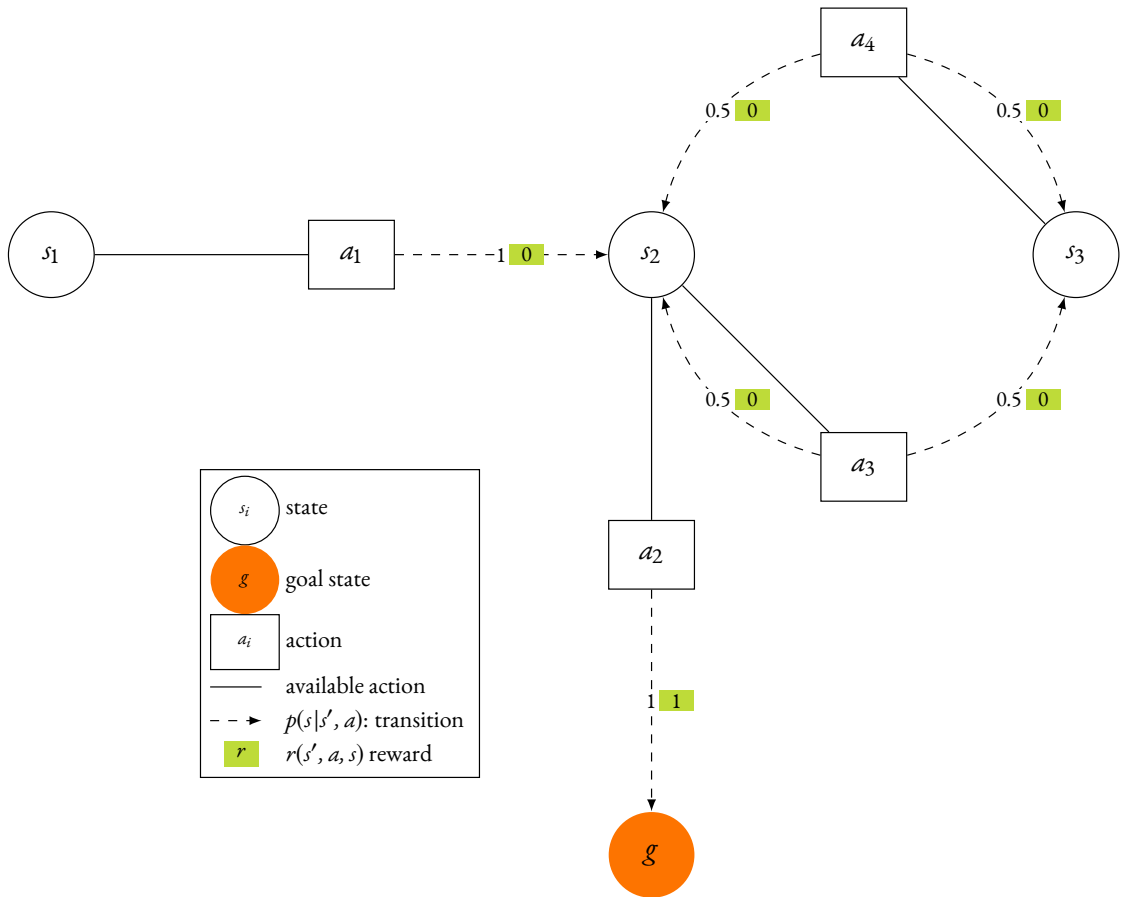


Figure 2.1: Example MAXPROB problem that is no Bertsekas-SSP

This thesis focuses on MDPs with an explicit set of goal-states. So, a *MAXPROB* is always considered with its originally specified set of goal-states. In the following, non-goal oriented MDPs are not considered any further and the relations of *MAXPROB* to the other classes with explicit goal states is investigated.

Observation 2.3.19:

$MAXPROB \not\subseteq Bertsekas-SSP$

The example MDP in Figure 2.1 is constructed by me and shows a *MAXPROB* problem that is no Bertsekas-SSP problem. The example MDP is a *MAXPROB* problem because only the transition to the goal state gives a reward of 1. All other rewards are 0. It is no Bertsekas-SSP problem since a policy that selects action a_3 in state s_2 is an improper policy. However, this policy would have a value of 0 and not $-\infty$.

However, it is possible to construct a *MAXPROB* problem that is a Bertsekas-SSP problem.

Observation 2.3.20:

$MAXPROB \cap Bertsekas-SSP \neq \emptyset$

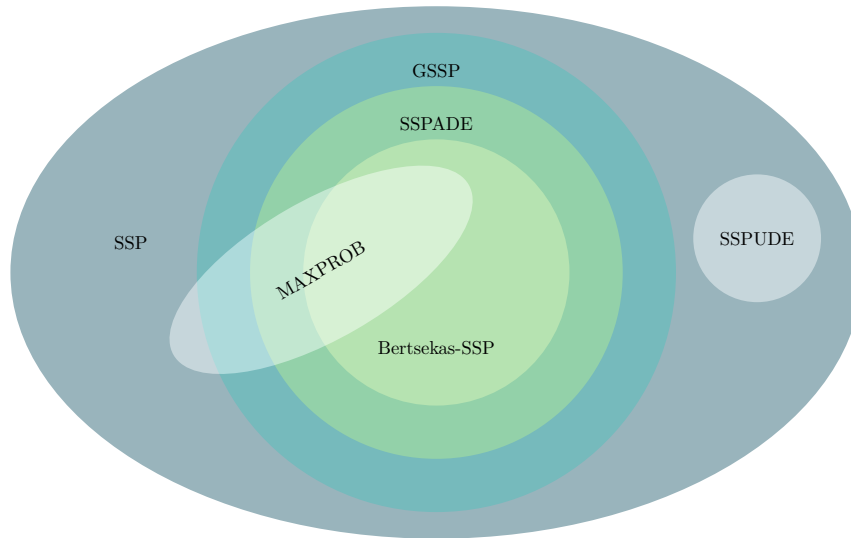


Figure 2.2: Hierarchy of MDP classes

Since in a MAXPROB problem all rewards are either 0 or 1, an improper policy in a MAXPROB problem can never have a value of $-\infty$. However, MAXPROB problems where only proper policies exist are Bertsekas-SSP problems. Consider for instance the MDP of Figure 2.1 without the actions a_3, a_4 and without the state s_3 . This is a MAXPROB problem where only one policy exists and this policy is proper.

Since a MAXPROB problem has only restrictions on the reward structure and does not require any proper policy, an MDP where there exists no transition to a goal state from any other state is a valid MAXPROB problem. This MAXPROB MDP has no proper policy. So, it can not be a GSSP problem.

Observation 2.3.21:

$$\text{MAXPROB} \not\subseteq \text{GSSP}$$

As mentioned above, in a MAXPROB problem, there exist only non-negative rewards. An SSPUDE is improper in s_1 . So if the problem contains actions, there must exist an improper policy with a reward of $-\infty$. This can not be possible in a MAXPROB problem because there every policy has a non-negative reward.

Observation 2.3.22:

$$\text{MAXPROB} \cap \text{SSPUDE} = \emptyset$$

Figure 2.2 summarizes the relations between the presented MDP classes without NEG MDPS and POSB MDPS.

2.3.2 Vector Notation

A finite state and action spaces is assumed throughout this thesis, so a vector notation can be used for MDPs. For $d \in D^{MD}$, the reward vector $r_d \in \mathbb{R}^n$ is defined as

$$(r_d)_s := r(s, d(s)) \forall s \in S$$

and analogous for $d \in D^{MR}$ as

$$(r_d)_s := \sum_{a \in A_s} q_{d(s)}(a) r(s, a) \forall s \in S.$$

The transition probabilities under a decision rule d can be captured in a transition probability matrix $P_d \in \mathbb{R}^{n \times n}$. For $d \in D^{MD}$, the (s, j) -th entry of P_d is defined as

$$(P_d)_{s,j} := p(j|s, d(s))$$

and for $d \in D^{MR}$ analogously as

$$(P_d)_{s,j} := \sum_{a \in A_s} q_{d(s)}(a) p(j|s, a).$$

Let $V = \mathbb{R}^n$ be the vector space of all possible values of the MDP where a component equals the value of the MDP in state s in S . A vector $v \in V$ will also be considered as the value function or value vector of the MDP. Then, evaluating a one-period MDP with terminal reward $r \in \mathbb{R}^n$ under decision rule d gives the value

$$r_d + P_d r.$$

It can be shown that under Assumption 2.3.1 of this section, the resulting vector is again in $V = \mathbb{R}^n$ (Puterman, 2005, Lemma 5.6.1).

The vector notation leads to a clearer representation of the policy evaluation introduced in Equation 2.1. The probability distribution that a system starting in s occupies in decision epoch $t + 1$ state j can be read off the transition matrix P_π^t which is calculated by multiplying $P_{d_1} \cdot P_{d_2} \cdot \dots \cdot P_{d_t}$. So,

$$(P_\pi^t)_{s,j} := [P_{d_1} \cdot P_{d_2} \cdot \dots \cdot P_{d_t}]_{s,j} = \mathbb{P}^\pi \{X_{t+1} = j \mid X_1 = s\}.$$

Given that the value of a policy v^π exists, it can be written in vector notation as

$$v^\pi = \sum_{t=1}^{\infty} (P_\pi^{t-1}) r_d. \quad (2.7)$$

2.3.3 Optimality Equations

The goal is to derive the optimality equations in vector notation for infinite-horizon MDPs under the expected total reward criterion. As in this subsection, the introduced vector notation is used to express the optimality equations for infinite-horizon MDPs under the expected total reward criterion, assume

that the limit of Equation 2.7 exists for all policies π and the value of the MDP is well-defined. Then, Equation 2.7 can be rewritten as

$$\begin{aligned} v^\pi &= \sum_{t=1}^{\infty} P_\pi^{t-1} r_d \\ &= r_{d_1} + P_{d_1}(r_{d_2} + P_{d_2}r_{d_3} + \dots) \\ &= r_{d_1} + P_{d_1}v^{\pi'}, \end{aligned}$$

where $\pi = (d_2, d_3, \dots)$. For a stationary, deterministic policy d^∞ this equation simplifies to

$$v^{d^\infty} = r_d + P_d v^{d^\infty},$$

so v^{d^∞} is a fixed point of the linear transformation $(r_d + P_d)(v)$. For MDPs under the expected *discounted* reward criterion, it can be shown that v^{d^∞} is the unique solution of $r_d + P_d$. However, this is not in general the case for MDPs under the expected total reward criterion.

We can write the optimality equations 2.4 of finite horizon MDPs as

$$v_t(s) = \max_{a \in \mathcal{A}_s} \left\{ r(s, a) + \sum_{j \in \mathcal{S}} p(j|s, a) v_{t+1}(j) \right\}, \quad \forall t \in T, \quad \forall s \in \mathcal{S}.$$

As we assume \mathcal{A}_s to be finite in each s , the maximum is attained in each state and we do not need a supremum in the stated equations. Applying the limit $t \rightarrow \infty$ to the last equations, we derive

$$v(s) = \max_{a \in \mathcal{A}_s} \left\{ r(s, a) + \sum_{j \in \mathcal{S}} p(j|s, a) v(j) \right\}, \quad \forall s \in \mathcal{S}. \quad (2.8)$$

We refer to equations 2.8 as the optimality equations for infinite-horizon MDPs. Equations 2.8 can be written in vector notation as

$$v = \max_{d \in D^{MR}} \{r_d + P_d v\}. \quad (2.9)$$

Note, that we have not said anything yet about the optimality equations. It remains to show under which conditions there exists a solution to the optimality equations for infinite-horizon MDPs under the expected total reward criterion and when this solution equals the value of the MDP.

As for finite-horizon MDPs, it can be shown that the maximum in Equation 2.9 only needs to be evaluated over all deterministic Markovian policies while attaining the same maximum value (Puterman, 2005, Prop. 6.2.1). For later use and further simplification, we define a linear operator, sometimes called dynamic programming operator or Bellmann operator, \mathcal{B} as

Definition 2.3.23 (Dynamic Programming Operator).

$$\mathcal{B}v := \max_{d \in D^{MD}} \{r_d + P_d v\}. \quad (2.10)$$

So, as a result we can summarize the optimality equations by

$$v = \mathcal{B}v. \quad (2.11)$$

A solution of the optimality equations is a fixed point of the dynamic programming operator \mathcal{B} .

2.3.4 Solutions of the Optimality Equations

In contrast to MDPs under the expected discounted reward criterion, the dynamic programming operator applied to total reward models is not necessarily a contraction mapping. For MDPs under the expected discounted reward criterion, this property of the dynamic programming operator makes it possible to apply the Banach fixed point theorem. The existence and uniqueness of a fixed point can then be followed from the Banach fixed point theorem. Furthermore, a sequence of value vectors derived from the dynamic programming operator starting with an arbitrary $v \in \mathbb{R}^n$ will converge to the unique fixed point. Together with the result that the fixed point of the dynamic programming operator equals the value of the considered discounted MDP, a basis for algorithms that yield an optimal policy for a discounted MDP is given. By examples, it will be shown that these conclusions can not generally be made for MDPs under the expected total reward criterion. Weaker results that hold for arbitrary expected total reward models are summarized. Furthermore, assumptions and special classes of expected total reward models are presented, where properties similar to those of discounted reward models hold.

From the definition of the value of an MDP, it can be shown that:

Theorem 2.3.24 (Puterman (2005), Thm. 7.1.3.):
The value v^ of an MDP under the total expected reward criterion satisfies the optimality equations 2.11.*

If one considers an MDP where only one decision rule d is available, the value of the MDP is obviously $v^* = v^{d^\infty}$. Applying Theorem 2.3.24 yields

$$v^{d^\infty} = \mathcal{B}v^{d^\infty} = \max_{d \in D^{MD}} \{r_d + P_d v^{d^\infty}\} = r_d + P_d v^{d^\infty}.$$

The dynamic programming operator \mathcal{B} of MDPs under the expected total reward criterion, defined in Definition 2.3.23, is a monotone function. Furthermore, if the value function is increased by a constant in each state, the result of the dynamic programming operator also increases in each state by this constant. The following Lemma shows these properties:

Lemma 2.3.25 (Puterman (2005), Lemma 7.1.5):

1. For all $u, v \in V$ with $u \leq v$, $\mathcal{B}u \leq \mathcal{B}v$.
2. For all $c \in \mathbb{R}$, $v \in V$, $\mathcal{B}(v + c\mathbb{1}) = \mathcal{B}v + c\mathbb{1}$.

PROOF. The proof of this Lemma uses that P_d is a transition matrix, where each row specifies a probability distribution.

1. Let $u, v \in V$ with $u \leq v$. Then, there exists a decision rule $d \in D^{MD}$ at which the maximum is attained, such that $\mathcal{B}u = r_d + P_d u$. It follows

$$\mathcal{B}u = r_d + P_d u \leq r_d + P_d v \leq \mathcal{B}v,$$

where the first inequality uses that P_d has only positive entries and u is component wise smaller than v .

2. Let $c \in \mathbb{R}$, $v \in V$.

$$\begin{aligned} \mathcal{B}(v + c\mathbb{1}) &= \max_{d \in D^{MD}} \{r_d + P_d(v + c\mathbb{1})\} \\ &= \max_{d \in D^{MD}} \{r_d + P_d v + cP_d \mathbb{1}\} \\ &= \max_{d \in D^{MD}} \{r_d + P_d v\} + c\mathbb{1} \\ &= \mathcal{B}v + c\mathbb{1}. \end{aligned}$$

In the second to last equation, it is used that for each $d \in D^{MD}$ the rows of P_d sum up to 1. ■

In contrast to discounted models, v^* is not a unique fixed point of \mathcal{B} . For any scalar $c \in \mathbb{R}$, $v^* + c\mathbb{1}$ is also a fixed point:

$$\mathcal{B}(v^* + c\mathbb{1}) \stackrel{\text{Lemma 2.3.25}}{=} \mathcal{B}v^* + c\mathbb{1} = v^* + c\mathbb{1}.$$

We summarize results of total reward MDP classes, where the fixed point v^* is unique or can be characterized:

- For Bertsekas-SSP MDPs, Bertsekas shows in Bertsekas, 2001, Prop. 2.1.2 that the dynamic programming operator \mathcal{B} has at most one fixed point. The proof does not rely on the Banach fixed point theorem. It uses that the value of a proper policy v^{d^∞} is the unique solution of

$$v = r_d + P_d v.$$

This in turn is a consequence of Equation 2.6 that shows that for proper policies the probability for not reaching a goal state after n steps goes to zero as $n \rightarrow \infty$. Since in a goal state no further reward is accumulated we get

$$\lim_{k \rightarrow \infty} P_d^k v = 0, \forall v \in V.$$

So, the reward from using a deterministic decision rule d infinitely often converges to the value v^{d^∞} of the stationary deterministic policy for every $v \in V$:

$$\sum_{m=0}^{k-1} P_d^m r_d + P_d^k v \xrightarrow{k \rightarrow \infty} v^{d^\infty}.$$

Therefore, if any other value function v' satisfies $v' = r_d + P_d v'$, we can recursively insert the equation for v' and get $v' = \lim_{k \rightarrow \infty} \sum_{m=0}^{k-1} P_d^m r_d + P_d^k v'$ which converges to v^{d^∞} . So v' must be v^{d^∞} and the solution of $v = r_d + P_d v$ is unique.

Finally, Bertsekas shows that the dynamic programming operator has a unique solution by using the uniqueness of $v = r_d + P_d v$ in the following way: Assume two fixed points \bar{v} and v' of \mathcal{B} . For Bertsekas-SSP MDP with a finite action set, there must exist two proper decision rules \bar{d} and d' at which the maximum of \mathcal{B} is attained. From the uniqueness of the fixed point of $v = r_d + P_d v$, it follows that $\bar{v} = v^{\bar{d}^\infty}$ and $v' = v^{d'^\infty}$.

We have $v = \mathcal{B}v = \mathcal{B}^k v \forall k \geq 1$ and all fixed points v . From that we get

$$\bar{v} \geq \lim_{k \rightarrow \infty} \sum_{m=0}^{k-1} P_{d'}^m r_{d'} + P_{d'}^k v = v^{d'^{\infty}} = v'$$

and analogous $v' \geq \bar{v}$. So the fixed point of \mathcal{B} must be unique.

As a side note, there exists a special case of Bertsekas-SSP MDPs where \mathcal{B} is a contraction in terms of a weighted supremum norm. This is the case if all policies are proper (Bertsekas, 2001, Ex. 2.14).

- Let $V^+ \subseteq V$ be the set of positive bounded value functions. For positive bounded models, it can be shown that v^* is the component-wise minimal fixed point of \mathcal{B} in V^+ (Puterman, 2005, Thm. 7.2.3). When evaluating a stationary deterministic policy d^∞ , the value v^{d^∞} of this policy is also the component-wise minimal solution of

$$v = r_d + P_d v$$

in V^+ for positive bounded models.

- Let $V^- \subseteq V$ be the set of negative bounded value functions. For negative models, it can be shown that v^* is the component-wise maximal fixed point of \mathcal{B} in V^- (Puterman, 2005, Thm. 7.3.3). When evaluating a stationary deterministic policy d^∞ , the value v^{d^∞} of this policy is also the component-wise maximal solution of

$$v = r_d + P_d v$$

in V^- for negative models.

In discounted MDPs, it is easy to identify an optimal decision rule. If a decision rule d^* is *conserving* which means

$$r_{d^*} + P_{d^*} v^* = v^*,$$

it follows that the deterministic stationary policy $(d^*)^\infty$ is an optimal decision rule (Puterman, 2005, Thm. 6.2.7). However, in MDPs under the expected total reward criterion, it is only a necessary not a sufficient condition that d is conserving.

Example 2.1:

Figure 2.3 shows an example from (Puterman, 2005, Ex. 7.2.3) with two states s_1 and s_2 . In state s_1 , there are two actions a_1 and a_2 available, in state s_2 , there is one action a_3 available. There exist two deterministic decision rules. Let d_1 be the decision rule that chooses action a_1 in s_1 and d_2 be the decision rule that chooses action a_2 in s_1 . Both decision rules select a_3 in s_2 .

The transition matrices under these decision rules are

$$P_{d_1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad P_{d_2} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}.$$

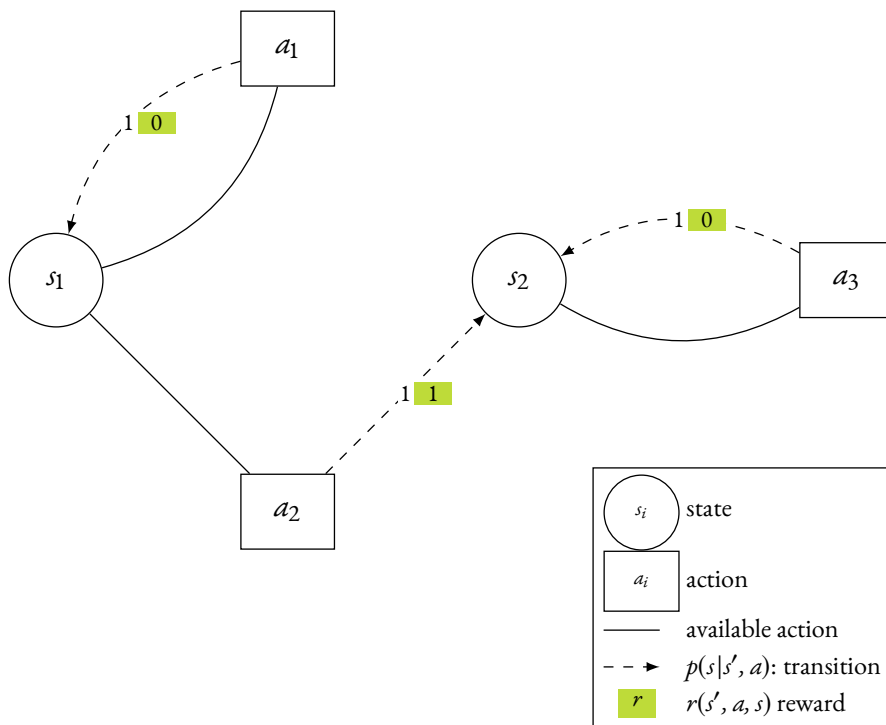


Figure 2.3: Conserving decision rule is not sufficient (Puterman, 2005, Ex. 7.2.3)

Since the presented MDP is a positive bounded MDP, we can calculate the value of the MDP by finding a minimal solution $v \in V^+$ of the optimality equations. The optimality equations for the example are

$$v(s_1) = \max\{1 + v(s_2), v(s_1)\}$$

$$v(s_2) = v(s_2).$$

The minimal solution of these equations in V^+ is

$$v^*(s_1) = 1, v^*(s_2) = 0.$$

Obviously, in this small example d_2 is optimal and d_1 not. This can be seen by computing the value of those decision rules, which are $v^{d_2^\infty}(s_1) = 1$ and $v^{d_1^\infty}(s_1) = 0$. However, both decision rules are conserving. We show this by evaluating $v^* = r_d + P_d v^*$ for both decision rules:

$$d_1 : \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$d_2 : \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad *$$

Since requiring a conserving decision rule is not a sufficient criterion, a second condition is needed. Theorem 2.3.26 (Puterman, 2005, Thm. 7.1.7) characterizes an optimal decision rule in MDPs under the expected total reward criterion:

Theorem 2.3.26 (Puterman (2005), Thm. 7.1.7):

If $d \in D^{MD}$ satisfies

$$r_d + P_d v^* = v^* \text{ and } \lim_{N \rightarrow \infty} \sup \mathbb{E}_s^{d^\infty} \{v^*(X_{N+1})\} \leq 0,$$

then d^∞ is optimal.

A decision rule that satisfies the second condition of Theorem 2.3.26 is called *equalizing*. This condition can be rewritten as $\lim_{N \rightarrow \infty} \sup P_d^N v^*(s) \leq 0$ for all $s \in S$ in MDPs under the expected total reward criterion. In discounted models, the discount factor λ guarantees that each decision rule is equalizing. There are classes of expected total reward models, where the same result holds:

- Since in negative models the value of all stationary policies and therefore also the value of the MDP is always less or equal zero Puterman, 2005, Prop. 7.3.1.

$$\lim_{N \rightarrow \infty} \sup P_d^N v^* \leq 0$$

is satisfied for all decision rules. So, in negative models all decision rules are equalizing.

- For proper decision rules

$$\lim_{N \rightarrow \infty} \sup P_d^N v = 0$$

holds for all $v \in V$ and therefore every proper decision rule is equalizing. Since in Bertsekas-SSP MDPs all improper policies have a reward of $-\infty$ and there exists at least one proper policy, it can be concluded that every policy that is conserving in a Bertsekas-SSP MDP is an optimal policy.

In positive bounded models, there may exist conserving decision rules that are not equalizing.

Example 2.1 (continued):

Revisiting Example 2.1, we see that d_2 is equalizing, while d_0 is not. The transition matrix for N steps are

$$P_{d_0}^N = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad P_{d_2}^N = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix},$$

from which $P_{d_0}^N v^* = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $P_{d_2}^N v^* = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ follows for all N . So,

$$\lim_{N \rightarrow \infty} \sup P_{d_2}^N v^*(s) = 0 \quad \forall s \in S$$

holds, and d_2 is equalizing. We can apply Theorem 2.3.26 to identify d_2 as an optimal stationary deterministic policy. *

After identifying a stationary deterministic optimal policy, the question is open whether there always exists a stationary deterministic optimal policy. Puterman relies on the proof for discounted models and uses a non-decreasing sequence of discount factors converging to 1 to prove the existence of a stationary deterministic optimal policy. For completeness, we restate the assumptions made in this section.

Theorem 2.3.27 (Puterman (2005), Thm. 7.1.9):

Suppose an MDP under the expected total reward criterion with a finite state and actions space that satisfies Assumption 2.3.2. Then, there exists a stationary deterministic optimal policy.

For Bertsekas-SSP MDPs, the existence of a stationary deterministic optimal policy can be proved without relying on discounted models (Bertsekas, 2001, Prop. 2.1.2).

2.3.5 Value Iteration

Value iteration is based on the convergence of the sequence $\{v^n\}$ defined by

$$v^{n+1} = \mathcal{B}v^n$$

to the value of the MDP v^* . In discounted models, this convergence is guaranteed for any $v \in \mathcal{V}$ by the Banach fixed point theorem. In MDPs under the expected total reward criterion, the convergence of $\{v^n\}$ does not hold in general.

Example 2.2:

Consider again the MDP of Figure 2.3. We choose $v^0 = (3, 2)^T \in \mathcal{V}$ and define $v^{n+1} = \mathcal{B}v^n$. The next values of the sequence are

$$v^1 = \begin{pmatrix} \max\{1 + 2, 3\} \\ 2 \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$$

$$v^2 = \begin{pmatrix} \max\{1 + 2, 3\} \\ 2 \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}.$$

Since $(3, 2)^T$ is another fixed point of the optimality equations, the sequence will stay in $(3, 2)^T$. Obviously, this is not the value of the MDP. *

Algorithm 3 shows the general value iteration algorithm. If it converges to the value of the MDP v^* and the termination criterion $\|v^{t+1} - v^t\| < \epsilon$ is met, suboptimal error bounds of the returned solution v^{t+1} can be computed. However, the bounds are not as powerful as for discounted MDPs and additional knowledge is required. For instance for Bertsekas-SSPs, the expected number of steps until reaching a goal state needs to be known (Hansen, 2017).

Some results regarding the convergence of the value iteration algorithm for the presented MDP classes of Subsection 2.3.1 are:

- For Bertsekas-SSP MDPs, we have already seen that v^* is the unique fixed point of \mathcal{B} . The sequence $\{v^n\}$ generated by the dynamic programming operator converges to the value of the MDP for every $v \in \mathcal{V}$ (Bertsekas, 2001, Prop. 2.1.2). So, for Bertsekas-SSP MDPs value iteration converges for every value function $v \in \mathcal{V}$ to the value of the MDP.
- For positive bounded MDPs, value iteration converges if $0 \leq v^0 \leq v^*$ holds (Puterman, 2005, Cor. 7.2.13).
- In negative MDPs with a finite the state set S value iteration converges whenever $0 \geq v^0 \geq v^*$ (Puterman, 2005, Cor. 7.3.12).

Algorithm 3: Value Iteration

Data: Infinite-Horizon MDP $(S, \mathcal{A}, p(\cdot|s, a), r(s, a))$

- 1 Select $v^0 \in \mathcal{V}$ appropriate, $\epsilon > 0$;
- 2 $t \leftarrow 0$;
- 3 Compute $v^{t+1} = \mathcal{B}v^t$;
- 4 if $\|v^{t+1} - v^t\| < \epsilon$ then
- 5 | return v^{t+1} ;
- 6 else
- 7 | $t \leftarrow t + 1$;
- 8 | Go to line 3;
- 9 end

In general, value iteration may require an infinite number of iterations and each iteration has a running time of $O(mn^2)$ (Littman, Dean, and Kaelbling, 1995).

For Bertsekas-SSP MDPs that contain an optimal stationary deterministic decision rule whose transition probability graph is acyclic, it can be shown that value iteration will yield the value of the MDP after at most n steps (Bertsekas, 2001, Sec. 2.2.1). A *consistently improving policy* $d^{*\infty}$, is a policy that satisfies for all $i \in S$

$$\forall j \in S \text{ with } p(j|i, d^{*}(i)) > 0 \Rightarrow v^{*}(i) > v^{*}(j).$$

The transition probability graph of a consistently improving policy is acyclic. So, if there exists a consistently improving policy, it can be derived that value iteration terminates after finitely many iterations.

When applying the value iteration algorithm to discounted MDPs or Bertsekas-SSP MDPs, an optimal stationary deterministic policy $d^{*\infty}$ can be derived by just remembering the action at which the maximum of the dynamic programming operator was attained. However, since in MDPs under the total expected reward criterion only a conserving policy that is also equalizing is an optimal stationary deterministic policy there need to be further considerations done, when identifying an optimal policy.

2.3.6 Policy Iteration

The general policy evaluation algorithm, presented by Algorithm 4, relies on the existence of an optimal stationary deterministic decision rule and not, like the value iteration algorithm, on finding a fixed point. It terminates at an optimal stationary deterministic decision rule after a finite number of iterations

for discounted models with a finite state and action space (Puterman, 2005, Thm. 6.4.2). The proof is based on the property that the sequence of value functions generated in the policy evaluation step is monotonically increasing and that there exist only finitely many different deterministic stationary policies.

Also for Bertsekas-SSP MDPs, it can be shown that the new policy generated by the policy iteration is either strictly better than the current policy or an optimal policy (Bertsekas, 2001, Sec. 2.2). This can be seen as follows: Let d be the current decision rule with value v^{d^∞} and \bar{d} be the policy at which the maximum of $\mathcal{B}v^{d^\infty}$ is attained. Then

$$r_{\bar{d}} + P_{\bar{d}}v^{d^\infty} = \mathcal{B}v^{d^\infty} \geq r_d + P_dv^{d^\infty} = v^{d^\infty}.$$

By using the monotonicity of $r_{\bar{d}} + P_{\bar{d}}u \geq r_{\bar{d}} + P_{\bar{d}}v$, $\forall u \geq v$, $u, v \in V$ and the convergence of

$$\sum_{m=0}^{k-1} P_{\bar{d}}^m r_{\bar{d}} + P_{\bar{d}}^k v \xrightarrow{k \rightarrow \infty} v^{\bar{d}^\infty}, \forall v \in V$$

for proper decision rules, we get $v^{\bar{d}^\infty} \geq v^{d^\infty}$. So, either there exists an $i \in S$ such that $v^{\bar{d}^\infty}(i) > v^{d^\infty}(i)$ or we get $v^{\bar{d}^\infty} = \mathcal{B}v^{d^\infty}$ and \bar{d}^∞ is an optimal policy. The shown property implies, that the policy iteration algorithm terminates after a finite number of steps for Bertsekas-SSP MDPs.

For expected total reward MDPs, the policy evaluation, step 3 in Algorithm 4, may not be possible. This is the case if the selected policy d_0 is a policy with an infinite value $|v^{d_0^\infty}| = \infty$. For example in negative models, there may exist a stationary deterministic policy with $v^{d_0^\infty} = -\infty$. But even if the algorithm starts with a decision rule d_0 that has a finite value, the policy evaluation step may not have a unique solution.

Example 2.3:

Consider again the MDP presented by Figure 2.3. Assume, the decision rule $d_1(s_1) = a_1$ and $d_1(s_2) = a_3$ is selected as the initial decision rule in the policy iteration algorithm. Then the policy evaluation step would be to find a solution of

$$\begin{aligned} (I - P_{d_1})v &= r_{d_1} \\ \Leftrightarrow \left(I - \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right) v &= \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ \Leftrightarrow \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} v &= \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \end{aligned}$$

Any $v \in \mathbb{R}^2$ solves this system of equations. *

It appears reasonable to select the minimal solution of $v \in V^+$ in positive bounded models and in negative models the maximal solution which is automatically in $v \in V^-$. This would guarantee, in

step 3 of Algorithm 4, that the value of the stationary deterministic policy d_0^∞ is finite.

Algorithm 4: Policy Iteration – General

Data: Infinite-Horizon MDP $(S, \mathcal{A}, p(\cdot|s, a), r(s, a))$

- 1 Select $d_0 \in D^{MD}$;
- 2 $t \leftarrow 0$;
- 3 Obtain v^n by solving

$$(I - P_{d_n})v = r_{d_n} \quad (\text{Policy Evaluation})$$

- 4 Choose d_{n+1} to satisfy

$$d_{n+1} \in \arg \max_{d \in D^{MD}} \{r_d + P_d v^n\} \quad (\text{Policy Improvement})$$

setting $d_{n+1} = d_n$ when possible;

- 5 if $d_{n+1} = d_n$ then
 - 6 | return d_n ;
 - 7 else
 - 8 | Go to line 3;
 - 9 end
-

If the policy evaluation is well-defined and the policy improvement steps leads to a termination of the algorithm, a decision rule $d_{n+1} = d_n$ has been found that can not be improved further. From $d_{n+1} = d_n$ follows

$$\begin{aligned} d_n &\in \arg \max_{d \in D^{MD}} \{r_d + P_d v^n\} \\ \Rightarrow \max_{d \in D^{MD}} \{r_d + P_d v^n\} &= r_{d_n} + P_{d_n} v^n = v^n \\ \Rightarrow Bv^n &= v^n. \end{aligned}$$

The last equation shows, that if the policy iteration algorithm terminates, a fixed point of the optimality equation has been found. In discounted models and Bertsekas-SSP MDPs, there exists only a unique fixed point which is v^* and it follows that d_n is a conserving decision rule. As mentioned earlier, deterministic stationary policies d_n^∞ derived from a conserving policy d_n are optimal in discounted models and Bertsekas-SSP MDPs.

In total reward models, this conclusions can not be made. Even if we can correctly evaluate all policies, we can not ensure the algorithm to terminate at the value of the MDP. So, the derived stationary deterministic policy can be suboptimal. Example 2.4 from Puterman, 2005, Ex. 7.3.1 shows a negative model, for which the policy iteration algorithm terminates at a suboptimal policy. This issue results from the fact that the set of recurrent states changes for different policies.

Example 2.4:

Consider the NEG MDP presented in Figure 2.4. Let δ be a decision rule that selects action a_1 in s_1 and γ be a decision rule that selects action a_2 in s_1 . Both decision rules use a_3 in s_2 .

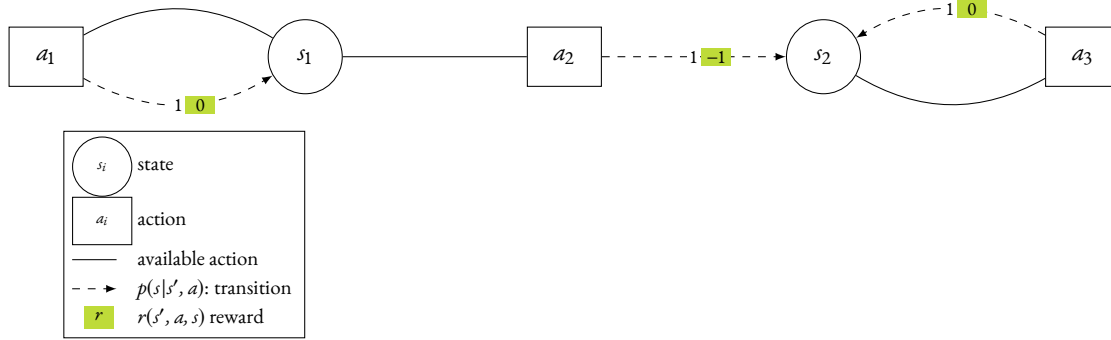


Figure 2.4: Policy Iteration may terminate with a suboptimal policy

Assume the policy iteration, Algorithm 4, starts with γ . According to our reasoning before, the maximal solution of $(I - P_{d_n})v = r_{d_n}$ is determined in the policy evaluation step such that we get a solution that equals the value of the considered policy. For our initial decision rule γ , the system of equations is

$$\begin{aligned}
 (I - P_\gamma)v &= r_\gamma \\
 \Leftrightarrow \left(I - \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \right) v &= \begin{pmatrix} -1 \\ 0 \end{pmatrix} \\
 \Rightarrow v(s_1) - v(s_2) &= -1 \wedge v(s_2) = 0 \\
 \Rightarrow \text{maximal solution: } v^0(s_1) &= -1, v^0(s_2) = 0
 \end{aligned}$$

At the policy improvement step, we determine the set of improving policies:

$$\begin{aligned}
 &\arg \max_{d \in D^{MD}} \{r_d + P_d v^0\} \\
 &= \arg \max \{r_\delta + P_\delta v^0, r_\gamma + P_\gamma v^0\} \\
 &= \arg \max \left\{ \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right\} \\
 &= \{\delta, \gamma\}.
 \end{aligned}$$

Since γ is in the set of improving policies, the policy algorithm terminates and returns γ . But $v^{\delta^\infty}(s_1) = 0 > -1 = v^{\gamma^\infty}(s_1)$, so γ is suboptimal. *

We have seen that there exist difficulties when defining a policy iteration algorithm for general expected total reward MDPs that terminates after a finite number of steps at an optimal deterministic stationary policy. Since positive bounded models play a role in the next chapters of this thesis, we present a modified policy iteration algorithm for positive bounded models in Algorithm 5. As mentioned earlier, the value v^{d^∞} of a stationary policy is the component-wise minimal solution of $v = r_d + P_d v$ in V^+ . So in the first step of the algorithm, a decision rule d_0 is chosen with $r_{d_0} \geq 0$. This choice of d_0 guarantees that the first value function v^0 and all further value functions v^n will be in V^+ . Another modification

occurs in step 3, where the minimal solution is computed. By computing the minimal solution and not an arbitrary solution, we know that the calculated value function equals $v^{d_n^\infty}$.

Algorithm 5: Policy Iteration - POSB MDPs

Data: Infinite-Horizon MDP $(S, \mathcal{A}, p(\cdot|s, a), r(s, a))$

- 1 Select $d_0 \in D^{MD}$ with $r_{d_0} \geq 0$;
- 2 $t \leftarrow 0$;
- 3 Obtain v^n by finding the minimal solution of

$$(I - P_{d_n})v = r_{d_n} \quad (\text{Policy Evaluation})$$

- 4 Choose d_{n+1} to satisfy

$$d_{n+1} \in \arg \max_{d \in D^{MD}} \{r_d + P_d v^n\} \quad (\text{Policy Improvement})$$

setting $d_{n+1} = d_n$ when possible;

- 5 if $d_{n+1} = d_n$ then
 - 6 | return d_n ;
 - 7 else
 - 8 | Go to line 3;
 - 9 end
-

Suppose the policy iteration algorithm has terminated. Then we have found a d_n that satisfies

$$d_n \in \arg \max_{d \in D^{MD}} \{r_d + P_d v^{d_n^\infty}\}.$$

In other words, $v^{d_n^\infty}$ satisfies $v^{d_n^\infty} = \mathcal{B}v^{d_n^\infty}$, i.e., it is a solution of the optimality equations. By definition of the value of an MDP $v^* \geq v^{d_n^\infty}$ holds. As stated in the last section, in POSB models v^* is the minimal solution of the optimality equations, so $v^* = v^{d_n^\infty}$ must be true.

We have seen that if the policy iteration algorithm for POSB models terminates, it terminates at the value of the POSB MDP and returns an optimal deterministic stationary policy. It remains to determine whether the algorithm terminates. Indeed, it can be shown that for finite-state MDPs, there exists a strict improvement in step 4.

Theorem 2.3.28 (Puterman (2005), Thm. 7.2.16):

Suppose a POSB MDP with finite state and action spaces. Let $\{v^n\}$ denote the sequence of value functions generated by Algorithm 5. Then for some finite N $v^N = v^$ and the returned d_n^∞ is optimal.*

In an MDP with n states and m actions, there exists n^m distinct deterministic, stationary policies, such policy iteration for SSO-MDP can take at most n^m iterations. The policy improvement step can be performed in $O(m \cdot n^2)$ and the policy evaluation step in $O(n^3)$ (Littman, Dean, and Kaelbling, 1995).

2.3.7 Linear Programming Formulation

For positive bounded models with finite state and action spaces, it is possible to formulate a linear program (LP), whose optimal value corresponds to the value of the POSB MDP. As stated above, the minimal solution $v \in V^+$ of $v = \mathcal{B}v$ equals the value of the MDP v^* . This is equivalent to

$$\begin{aligned} & \left| \begin{array}{l} \text{min. solution } v \\ \text{s.t. } v = \max_{d \in D^{MD}} \{r_d + P_d v\} \\ v \in V^+ \end{array} \right| \\ \Leftrightarrow & \left| \begin{array}{l} \text{min. solution } v \\ \text{s.t. } v \geq r_d + P_d v, \forall d \in D^{MD} \\ v \in V^+ \end{array} \right| \\ \Leftrightarrow & \left| \begin{array}{l} \text{min. solution } v \\ \text{s.t. } v(s) - \sum_{j \in S} p(j|s, a)v(j) \geq r(s, a), \forall a \in \mathcal{A}_s, \forall s \in S \\ v(s) \geq 0, \forall s \in S. \end{array} \right| \end{aligned}$$

By a minimal solution, a component-wise minimal solution is meant. So, the minimal solution can be computed by minimizing a positive weighted sum of the components. It can be shown, that the derived optimal decision rule from the linear program is independent of the choice of the weights as long as they are strictly positive (Puterman, 2005, Prop. 7.2.19). However, to give them an interpretation as a distribution over the state set, it will be assumed that the weights $\omega \in \mathbb{R}_+^S$ satisfy $\sum_{s \in S} \omega(s) = 1$. All together, the primal linear programming formulation for positive bounded models is:

$$\left| \begin{array}{l} \text{min } \sum_{s \in S} \omega(s)v(s) \\ \text{s.t. } v(s) - \sum_{j \in S} p(j|s, a)v(j) \geq r(s, a), \forall a \in \mathcal{A}_s, \forall s \in S \\ v(s) \geq 0, \forall s \in S. \end{array} \right| \quad (\text{primal LP})$$

The difference to a linear programming formulation for discounted models is that no discount factor occurs and that only solutions $v \in V^+$ are considered. The primal LP has $\sum_{s \in S} |\mathcal{A}_s|$ many inequalities and $|S|$ many variables with non-negativity constraints.

The primal LP formulation without the non-negativity conditions can also be applied to Bertsekas-SSP MDPs. As $\mathcal{B}^k v$ converges to v^* for all $v \in V$ and the dynamic programming operator \mathcal{B} is a monotone function, the following implication holds:

$$v \geq \mathcal{B}v \Rightarrow v \geq v^* = \mathcal{B}v^*.$$

So, the value of a Bertsekas-SSP MDP is also the minimal solution that satisfies $v \geq \mathcal{B}v$. The non-negativity conditions ensured $v \in V^+$ for POSB models. Since in Bertsekas-SSP MDPs the convergence of the dynamic programming operator to the value of the MDP holds for all $v \in V$ and v^* need not be an element of V^+ , the non-negativity conditions must be omitted in the primal LP formulation for Bertsekas-SSP MDPs.

Determining an optimal deterministic stationary decision rule of a POSB MDP by the primal LP is especially useful if there exists a single conserving decision rule. In that case, for exactly one action in each state the inequality $v(s) - \sum_{j \in S} p(j|s, a)v(j) \geq r(s, a)$ is satisfied with equality. The decision rule

that selects this actions is conserving. In expected total reward models, a decision rule must necessarily be conserving to be an optimal policy and, by Theorem 2.3.27, there always exists an optimal deterministic stationary policy. So, if the found decision rule is the only conserving decision rule, it must be an optimal decision rule. If there exist more than one conserving decision rule, it is necessary to establish which of them is also equalizing.

The dual linear program to the primal LP presented before equals:

$$\left| \begin{array}{l} \max \sum_{s \in S} \sum_{a \in A_s} r(s, a)x(s, a) \\ \text{s.t. } \sum_{a \in A_j} x(j, a) - \sum_{s \in S} \sum_{a \in A_s} p(j|s, a)x(s, a) \leq \omega(j), \forall j \in S \\ x(s, a) \geq 0, \forall a \in A_s, \forall s \in S. \end{array} \right| \quad (\text{dual LP})$$

For discounted models and Bertsekas-SSP MDPs, the primal linear program has no non-negativity condition $v(s) \geq 0$. Thus, the dual linear programming formulation for those MDPs has equalities instead of less or equal inequalities. This leads to a one-to-one relation between feasible solutions of the dual program and randomized decision rules for discounted MDPs and Bertsekas-SSP. Thereby, from a decision rule $d \in D^{MR}$ the feasible solution $x_d(s, a)$ to the dual program is defined as

$$x_d(s, a) := \sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \lambda^{n-1} \mathbb{P}^{d^\infty} \{X_n = s, Y_n = a \mid X_1 = j\}. \quad (2.12)$$

And the other way around, for a feasible solution $x(s, a)$ of the dual program, it can be shown that $\sum_{a' \in A_s} x(s, a') > 0$ holds for all $s \in S$ and a randomized decision rule d_x^∞ is well-defined by

$$\mathbb{P} \{d_x(s) = a\} = \frac{x(s, a)}{\sum_{a' \in A_s} x(s, a')} \forall s \in S. \quad (2.13)$$

Further, it can be shown that the corresponding $x_{d_x}(s, a)$ from the constructed decision rule d_x is feasible to the dual program and $x_{d_x}(s, a) = x(s, a)$ holds for all $a \in A_s, s \in S$ (Puterman, 2005, Thm. 6.9.1).

For POSB MDPs, the solution $x(s, a) = 0, \forall a \in A_s, \forall s \in S$ is always a feasible solution of the dual LP. So, due to the inequalities in the dual program the condition $\sum_{a' \in A_s} x(s, a') > 0$ need not hold for any $s \in S$. Example 2.5 shows that for POSB models, there exist multiple feasible dual solutions that correspond to the same decision rule.

Example 2.5:

Suppose the POSB MDP presented in Figure 2.5 with two states s_1 and s_2 . The state s_1 has two actions a_1 and a_2 available. The absorbing state s_2 has one available action which is a_3 . The dual linear program for this example is

$$\left| \begin{array}{l} \max x(s_1, a_1) + x(s_1, a_2) \\ \text{s.t. } x(s_1, a_1) + x(s_1, a_2) \leq \omega(s_1) \\ x(s_2, a_3) - x(s_1, a_1) - x(s_1, a_2) - x(s_2, a_3) \leq \omega(s_2) \\ x(s, a) \geq 0 \forall a \in A_s, \forall s \in S \end{array} \right|$$

Suppose the initial starting distribution is, e.g., $\omega(s_1) = 0.8$ and $\omega(s_2) = 0.2$. Then, for all $c \in \mathbb{R}_{\geq 0}$ the solution:

$$x(s_1, a_1) = 0.2$$

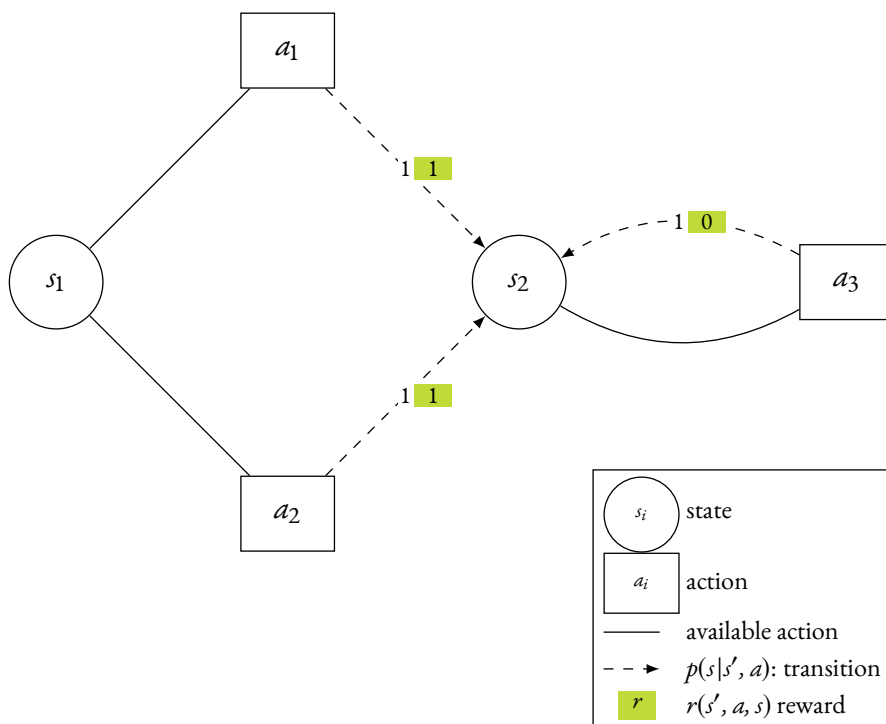


Figure 2.5: Many feasible dual solutions to a decision rule

$$\begin{aligned} x(s_1, a_2) &= 0.2 \\ x(s_2, a_3) &= c \in \mathbb{R}_{\geq 0}, \end{aligned}$$

is a feasible solution to the dual linear program. We can calculate a decision rule d from this feasible solution by using Equation 2.13 for states with $\sum_{a' \in \mathcal{A}_s} x(s, a') > 0$. For s_1 , we get that $\mathbb{P}\{d(s_1) = a_1\} = \mathbb{P}\{d(s_1) = a_2\} = 0.5$. In state s_2 , the variable $x(s_2, a_3) = c$ may be zero, but for all $x(s_2, a_3) > 0$, we get $d(s_2) = a_3$. So, there exist multiple feasible solutions of the dual program for a POSB MDP that lead to the same decision rule.

If we construct a feasible solution x_d following Equation 2.12 with $\lambda = 1$ from the calculated decision rule d , we obtain

$$\begin{aligned} x_d(s_1, a_1) &= 0.4 \\ x_d(s_1, a_2) &= 0.4 \\ x_d(s_2, a_3) &= \infty. \end{aligned}$$

Obviously, $x_d(s, a) = x(s, a)$ does not hold. However, the conducted decision rule from x_d according to Equation 2.13 equals d in state s_1 . *

As Puterman (2005) captures for POSB models only a statement about optimal basic solutions, which will be stated later, it will be figured out in this thesis how a decision rule can be derived from

any feasible solution. The feasible solution need not necessarily be a basic solution of the dual LP. Furthermore, it will be shown how the objective value of an arbitrary feasible solution is related to the expected total reward of the derived decision rule.

First, a decision rule derived from an arbitrary feasible solution of the dual LP is defined. This definition extends the prescription of 2.13, which defined a decision rule corresponding to a feasible solution of the dual linear program for discounted MDPs.

Definition 2.3.29 (Decision rule from feasible solution).

Let x be a feasible solution of the dual LP and $u(s) := \sum_{a' \in \mathcal{A}_s} x(s, a')$, $\forall s \in S$. We define a corresponding decision rule d_x as

$$\mathbb{P}\{d_x(s) = a\} = \begin{cases} \frac{x(s,a)}{u(s)} & \forall s \in S \text{ with } u(s) > 0 \\ q \in \mathcal{P}(\mathcal{A}_s) & \text{else.} \end{cases}$$

In the case where $u(s) = 0$ holds, an arbitrary probability distribution can be chosen. Let $S^* \subset S$ be the set of states where $u(s) > 0$ holds. If $S = S^*$ holds, Definition 2.3.29 defines a unique d_x . But, if there exists an $s \in S \setminus S^*$, there exist infinitely many decision rules d_x corresponding to x following Definition 2.3.29. Before we show the relationship between the objective value of x and $v^{d_x^*}$, we define two LPs that are similar to the primal LP and the dual LP but correspond to an MDP where only one decision rule d is available. Let LP_d be a primal linear program for a POSB MDP where only one decision rule d is available:

$$\left| \begin{array}{l} \min \omega^T v \\ \text{s.t. } v - P_d v \geq r_d \\ v \geq 0. \end{array} \right| \quad (LP_d)$$

The linear program LP_d has $|S|$ many inequalities and $|S|$ many variables with non-negativity constraints. Its dual linear program is denoted by DP_d and equals:

$$\left| \begin{array}{l} \max r_d^T x \\ \text{s.t. } x - P_d^T x \leq \omega \\ x \geq 0. \end{array} \right| \quad (DP_d)$$

Recap from Subsection 2.3.2 the definition of $P_d \in \mathbb{R}^{n \times n}$ as the transition matrix under decision rule d and the definition of $r_d \in \mathbb{R}^n$ as the expected reward from using decision rule d . The LPs LP_d and DP_d are special variants of primal LP and dual LP where only one decision rule d exists. From earlier investigations of Subsection 2.3.3 or from the fact that in the case of a POSB MDP with one decision rule v^* equals the value of that decision rule, we know that the optimal solution of LP_d equals v^{d^*} .

The following Theorem 2.3.30 fills the mentioned gap in Puterman (2005) by examining the relationship between the objective value of a feasible solution of the dual LP (the general dual LP including all decision rules) and the value of the derived decision rule d_x of that feasible solution x .

Theorem 2.3.30:

Let x be a feasible solution of the dual LP and d_x be the decision rule derived from x following Definition 2.3.29. Then:

1. $u(s) := \sum_{a' \in \mathcal{A}_s} x(s, a')$, $\forall s \in S$ is feasible for DP_{d_x} .

2. If d_x is a decision rule with a non-negative value, then

$$\sum_{s \in S} \sum_{a \in \mathcal{A}_s} r(s, a) x(s, a) \leq \omega^T v^{d_x^\infty}.$$

PROOF. 1. Obviously, $u(s) \geq 0$ holds for all $s \in S$. We show that the other inequalities are also satisfied. Let $S^* := \{s \in S \mid u(s) > 0\}$ and observe that from $u(s) = 0$ and the non-negativity of x , it follows that $x(s, a) = 0$, $\forall a \in \mathcal{A}_s$. Let $j \in S$, then

$$\begin{aligned} & u(j) - \left(P_{d_x}^T\right)_{j*} u \\ &= \sum_{a \in \mathcal{A}_j} x(j, a) - \sum_{s \in S} (P_{d_x})_{s,j} u(s) \\ &= \sum_{a \in \mathcal{A}_j} x(j, a) - \sum_{s \in S} \left[\sum_{a \in \mathcal{A}_s} q_{d_x(s)}(a) p(j|s, a) \right] u(s) \\ &= \sum_{a \in \mathcal{A}_j} x(j, a) - \sum_{s \in S^*} \left[\sum_{a \in \mathcal{A}_s} \frac{x(s, a)}{u(s)} p(j|s, a) u(s) \right] - \sum_{s \in S \setminus S^*} \left[\sum_{a \in \mathcal{A}_s} q_{d_x(s)}(a) p(j|s, a) \right] u(s) \\ &= \sum_{a \in \mathcal{A}_j} x(j, a) - \sum_{s \in S^*} \sum_{a \in \mathcal{A}_s} p(j|s, a) x(s, a) - \underbrace{\sum_{s \in S \setminus S^*} \left[\sum_{a \in \mathcal{A}_s} q_{d_x(s)}(a) p(j|s, a) u(s) \right]}_{=0} \\ &= \sum_{a \in \mathcal{A}_j} x(j, a) - \sum_{s \in S^*} \sum_{a \in \mathcal{A}_s} p(j|s, a) x(s, a) - \underbrace{\sum_{s \in S \setminus S^*} \sum_{a \in \mathcal{A}_s} p(j|s, a) x(s, a)}_{=0} \\ &= \sum_{a \in \mathcal{A}_j} x(j, a) - \sum_{s \in S} \sum_{a \in \mathcal{A}_s} p(j|s, a) x(s, a) \leq \omega(j). \end{aligned}$$

The last inequality follows since x is a feasible solution of the dual LP. Hence, we have shown

$$u(j) - \left(P_{d_x}^T\right)_{j*} u \leq \omega(j), \quad \forall j \in S$$

and u is feasible for DP_d .

2. We prove this statement by showing that

$$\sum_{s \in S} \sum_{a \in A_s} r(s, a)x(s, a) = \sum_{s \in S} r_{d_x}(s)u(s),$$

where u is defined as in the first statement of this theorem. Since $v^{d_x^\infty} \geq 0$ holds by assumption, $v^{d_x^\infty}$ is feasible for LP_d with $d = d_x$. By the first part of this theorem, u is feasible for DP_d with $d = d_x$. So, we can follow from weak duality that $r_{d_x}^T u \leq \omega^T v^{d_x^\infty}$ holds and hence

$$\sum_{s \in S} \sum_{a \in A_s} r(s, a)x(s, a) \leq \omega^T v^{d_x^\infty}$$

is shown. It is left to show the equality of the objective functions

$$\begin{aligned} & \sum_{s \in S} \sum_{a \in A_s} r(s, a)x(s, a) \\ & \stackrel{x(s,a)=0, \forall a \in A_s, \forall s \in S \setminus S^*}{=} \sum_{s \in S^*} \sum_{a \in A_s} r(s, a) \frac{x(s, a)}{u(s)} u(s) \\ & = \sum_{s \in S^*} u(s) \sum_{a \in A_s} r(s, a)q_{d_x(s)}(a) \\ & = \sum_{s \in S^*} u(s)r_{d_x}(s) \\ & \stackrel{u(s)=0, \forall s \in S \setminus S^*}{=} \sum_{s \in S} u(s)r_{d_x}(s) \\ & = r_{d_x}^T u. \end{aligned}$$

As argued above the statement follows from weak duality. ■

We can easily construct an example where the value of the objective function is strictly smaller than the value of the corresponding decision rule.

Example 2.5 (continued):

The vector $x(s, a) = 0, \forall a \in A_s, \forall s \in S$ is feasible for the dual program of any POSB MDP. The objective value of that solution is 0. When constructing a decision rule following Definition 2.3.29 according to this feasible solution, we can choose at each state $s \in S$ an arbitrary probability distribution $q_{d_x(s)}$. If there exists any decision rule d_x in the MDP with a strictly positive value, we get

$$0 = \sum_{s \in S} \sum_{a \in A_s} r(s, a)x(s, a) < \sum_{s \in S} \omega(s)v^{d_x^\infty}(s).$$

In the POSB MDP presented in Figure 2.5, we could for instance choose $d_x(s_1) = a_1$ and $d_x(s_2) = a_3$, which has a value of 1. *

The following Theorem 2.3.31 was developed in the connection with this thesis. It derives a unique feasible solution x_d for the dual LP from a decision rule d . If $x_d(s)$ is finite for all $s \in S$, the objective value of x_d equals the value of the stationary policy d^∞ .

Theorem 2.3.31:

Let d be a decision rule in a POSB MDP whose value v^{d^∞} is non-negative. Define

$$x_d(s, a) := \sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \mathbb{P}^{d^\infty} \{X_n = s, Y_n = a \mid X_1 = j\}. \quad (2.14)$$

and $x_d(s) := \sum_{a \in A_s} x_d(s, a)$. Then if $x_d(s) < \infty$ for all $s \in S$, it holds:

1. $x_d(s)$ is feasible for DP_d and satisfies all inequalities with equality.
2. $x_d(s, a)$ is feasible for dual LP.
3. The objective value of x_d in dual LP satisfies

$$\sum_{s \in S} \sum_{a \in A_s} r(s, a) x_d(s, a) = \sum_{s \in S} \omega(s) v^{d^\infty}(s).$$

4. x_d is an optimal solution of DP_d .

PROOF. I. Obviously, $x_d(s) \geq 0$, $\forall s \in S$ holds.

It has to be shown that $x_d - P_d^T x_d = \omega$ holds. Let $s \in S$ be an arbitrary state, then

$$\begin{aligned} x_d(s) &= \sum_{a \in A_s} x_d(s, a) \\ &= \sum_{a \in A_s} \left[\sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \mathbb{P}^{d^\infty} \{X_n = s, Y_n = a \mid X_1 = j\} \right] \\ &= \sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \sum_{a \in A_s} \mathbb{P}^{d^\infty} \{X_n = s, Y_n = a \mid X_1 = j\} \\ &= \sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \mathbb{P}^{d^\infty} \{X_n = s \mid X_1 = j\} \\ &= \sum_{j \in S} \omega(j) \left[\sum_{n=1}^{\infty} \mathbb{P}^{d^\infty} \{X_{n+1} = s \mid X_1 = j\} + \mathbb{P}^{d^\infty} \{X_1 = s \mid X_1 = j\} \right] \\ &= \sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \mathbb{P}^{d^\infty} \{X_{n+1} = s \mid X_1 = j\} + \omega(s) \\ &= \sum_{j \in S} \omega(j) \left[\sum_{k \in S} \sum_{a \in A_k} q_d(k)(a) p(s|k, a) \sum_{n=1}^{\infty} \mathbb{P}^{d^\infty} \{X_n = k, Y_n = a \mid X_1 = j\} \right] + \omega(s) \end{aligned} \quad (2.15)$$

$$\begin{aligned}
&= \sum_{k \in S} \sum_{a \in \mathcal{A}_k} q_{d(k)}(a) p(s|k, a) \left[\sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \mathbb{P}^{d^\infty} \{X_n = k, Y_n = a \mid X_1 = j\} \right] + \omega(s) \\
&= \sum_{k \in S} p_d(s|k) \sum_{a \in \mathcal{A}_k} x_d(k, a) + \omega(s) \\
&= \sum_{k \in S} p_d(s|k) x_d(k) + \omega(s).
\end{aligned}$$

The terms of the infinite sum can be rearranged, like e.g. in line 2.15, since by assumption the limit is finite and only non-negative terms appear. Therefore, it is an absolute convergent series and a reordering does not change the limit of the series. We have shown that x_d is feasible for DP_d and satisfies all inequalities of DP_d with equality. In an example following this theorem, it is shown that this result does not hold if there exist an $s \in S$ with $x_d(s) = \infty$.

2. Obviously, $x_d(s, a) \geq 0$ holds for all $a \in \mathcal{A}_s$, $s \in S$. We can use that x_d is feasible for DP_d to proof the other inequalities:

$$\begin{aligned}
\sum_{a \in \mathcal{A}_s} x_d(s, a) &= x_d(s) \\
&= \sum_{k \in S} p_d(s|k) x_d(k) + \omega(s) \\
&= \sum_{k \in S} \sum_{a \in \mathcal{A}_k} q_{d(k)}(a) p(s|k, a) x_d(k) + \omega(s) \\
&= \sum_{k \in S} \sum_{a \in \mathcal{A}_k} p(s|k, a) x_d(k, a) + \omega(s), \quad \forall s \in S.
\end{aligned}$$

We again used from above that

$$x_d(k) = \sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \mathbb{P}^{d^\infty} \{X_n = k \mid X_1 = j\}$$

and hence $q_{d(k)}(a) \cdot x_d(k) = x_d(k, a)$. So, $x_d(k, a)$ is also feasible for the dual LP.

3. We first show that

$$\sum_{s \in S} r_d(s) x_d(s) = \sum_{s \in S} \omega(s) v^{d^\infty}(s).$$

We have already shown that $x_d(s)$ is feasible for DP_d and satisfies

$$x_d = P_d^T x_d + \omega.$$

We can insert x_d recursively in this equation and get

$$x_d = \sum_{n=0}^{N-1} \left(P_d^T\right)^n \omega + \left(P_d^T\right)^N x_d.$$

We transpose this equation and multiply it from right with r_d :

$$x_d^T r_d = \omega^T \sum_{n=0}^{N-1} (P_d)^n r_d + x_d^T (P_d)^N r_d.$$

Next, we take the limit $N \rightarrow \infty$. Since by the assumption of a POSB model $v_+^{d^\infty} < \infty$ holds together with the assumption of this theorem that d^∞ has a non-negative value, we know that the first term on the right hand side converges to the finite value $\omega^T v^{d^\infty}$. It is a necessary condition for $\sum_{n=0}^{N-1} (P_d)^n r_d$ to be a convergent series that the sequence of $(P_d)^n r_d$ is a null sequence. Hence, the second term on the right hand side of the last equation converges to zero. All in all we get:

$$x_d^T r_d = \omega^T v^{d^\infty}.$$

It remains to show that the objective value of x_d in dual LP is identical to the objective value of x_d in DP_d :

$$\begin{aligned} & \sum_{s \in S} \sum_{a \in A_s} r(s, a) x_d(s, a) \\ &= \sum_{s \in S} \sum_{a \in A_s} r(s, a) \left[\sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \mathbb{P}^{d^\infty} \{X_n = s, Y_n = a \mid X_1 = j\} \right] \\ &= \sum_{s \in S} \sum_{a \in A_s} r(s, a) \left[\sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \mathbb{P}^{d^\infty} \{Y_n = a \mid X_n = s, X_1 = j\} \mathbb{P}^{d^\infty} \{X_n = s \mid X_1 = j\} \right] \\ &= \sum_{s \in S} \sum_{a \in A_s} r(s, a) \left[\sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} q_{d(s)}(a) \mathbb{P}^{d^\infty} \{X_n = s \mid X_1 = j\} \right] \\ &= \sum_{s \in S} \sum_{a \in A_s} r(s, a) q_{d(s)}(a) \left[\sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \mathbb{P}^{d^\infty} \{X_n = s \mid X_1 = j\} \right] \\ &= \sum_{s \in S} r_d(s) \left[\sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \mathbb{P}^{d^\infty} \{X_n = s \mid X_1 = j\} \right] \\ &= \sum_{s \in S} r_d(s) x_d(s). \end{aligned}$$

4. By weak duality $r_d^T x \leq \omega^T v^{d^\infty}$ holds for all x feasible for DP_d . Since x_d is feasible for DP_d and $r_d^T x_d = \omega^T v^{d^\infty}$, it must be an optimal solution. \blacksquare

The assumption $x_d(s) < \infty, \forall s \in S$ is crucial such that Theorem 2.3.31 holds:

Example 2.6:

Consider again the POSB MDP presented in Figure 2.5. We define a decision rule $d(s_1) = a_1$ and $d(s_2) = a_3$ together with a start distribution $\omega(s_1) = 0.2$, $\omega(s_2) = 0.8$. Under this decision rule $x_d(s_2, a_3) = x_d(s_2) = \infty$ as seen before. So the assumption of Theorem 2.3.31 does not hold.

The DP_d corresponding to the defined decision equals

$$\left| \begin{array}{l} \max x(s_1) \\ \text{s.t. } x(s_1) \leq 0.2 \\ -x(s_1) \leq 0.8 \\ x(s) \geq 0 \forall s \in S \end{array} \right|$$

From the first inequality and the non-negativity constraints, it follows that $x(s_1) \in [0, 0.2]$ and hence the second inequality can not be satisfied with equality. *

The reader may notice that in a POSB MDP with a finite number of action and states, there will always exist at least one $s \in S$ with $x_d(s)$ not being finite. This can be seen as follows: As we have a closed system no probability mass can be lost or generated and at each point in time t a probability mass of 1 develops from one state to another, i.e.,

$$\sum_{i \in S} \sum_{j \in S} \mathbb{P}^{d^\infty} \{X_t = j, X_{t+1} = i\} = 1, \forall t \in T.$$

And as we consider an infinite horizon, we get in total an infinite probability mass that is divided up by Equation 2.14 on finitely many variables $x_d(s, a)$. So there must exist at least one $x_d(s, a)$ which is infinite and the condition $x_d(s) < \infty$ can not hold for all $s \in S$.

However, it is possible to allow $x_d(s)$ to be unbounded on some subset of states \tilde{S} such that

$$\sum_{s \in S} \sum_{a \in A_s} r(s, a) x_d(s, a) = \sum_{s \in S} \omega(s) v^{d^\infty}(s)$$

still holds. Define $\tilde{S} := \{s \in S \mid x_d(s) = \infty\}$. If we go through the proof, we see that it still holds if all $s \in \tilde{S}$ satisfy that

- $s \in \tilde{S}$ is reward free, i.e., $r(s, a) = 0, \forall a \in A_s$,
- $s \in \tilde{S}$ is absorbing, i.e., $p(s|s, a) = 1 \forall a \in A_s$,
- and the start-distribution is 0, i.e., $\omega(s) = 0, \forall s \in \tilde{S}$.

In part I of Theorem 2.3.31, the equation

$$x_d(s) = \sum_{k \in S} p_d(s|k) x_d(k) + \omega(s)$$

still holds for all $s \in S \setminus \tilde{S}$ as $\sum_{k \in S} p_d(s|k) x_d(k) = \sum_{k \in S \setminus \tilde{S}} p_d(s|k) x_d(k) < \infty$ and we still have an absolute convergent series that can be rearranged. For all $s \in \tilde{S}$, the inequality becomes

$$- \sum_{k \in S \setminus \tilde{S}} p_d(s|k) x_d(k) \leq 0$$

since s is absorbing and $\omega(s) = 0$ for all $s \in \tilde{S}$. So, $x_d(s)$ is feasible for DP_d and for all $s \in S \setminus \tilde{S}$ the inequalities are satisfied with equality.

The second part of Theorem 2.3.31 holds by the same arguments as part 1.

The third and the fourth part stays valid since

$$\sum_{s \in S} r_d(s)x_d(s) = \sum_{s \in S \setminus \tilde{S}} r_d(s)x_d(s) = \sum_{s \in S \setminus \tilde{S}} \omega(s)v^{d^\infty}(s) = \sum_{s \in S} \omega(s)v^{d^\infty}(s).$$

Note that in Example 2.6, $\tilde{S} = \{s_2\}$ but $\omega(s_2) = 1$. So in that example the required properties of \tilde{S} do not hold.

Furthermore, the reader may notice that we end up with a Bertsekas-SSP MDP. The reward free states correspond to the goal states. And as in POSB MDPs $v_+^{d^\infty} < \infty$ holds together with the assumption of this theorem that v^{d^∞} is non-negative, we are particular considering proper policies in this Theorem 2.3.31.

Some observations can be made about the feasible solution x_d derived from a decision rule d , see Equation 2.14. In all states s with $x_d(s) = 0$, it can be concluded that the Markov chain derived from using decision rule d will not reach that state.

Proposition 2.3.32:

Let x_d be a the vector computed from decision rule d by Equation 2.14. Define $S^* := \{s \in S : \sum_{a \in A_s} x_d(s, a) > 0\}$, then the Markov chain derived from using decision rule d will not reach the states in $S \setminus S^*$.

PROOF. Let s be a state in $S \setminus S^*$. Then $\sum_{a \in A_s} x_d(s, a) = 0$. Using the non-negativity of $x_d(s, a)$, we can conclude that $x_d(s, a) = 0, \forall a \in A_s$. By using Equation 2.14, we can follow

$$\sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \mathbb{P}^{d^\infty} \{X_n = s, Y_n = a \mid X_1 = j\} = 0, \forall a \in A_s.$$

The probability that a Markov chain according to decision rule d reaches state s starting with an initial starting distribution ω equals

$$\begin{aligned} & \sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \mathbb{P}^{d^\infty} \{X_n = s \mid X_1 = j\} \\ &= \sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \sum_{a \in A_s} \mathbb{P}^{d^\infty} \{X_n = s, Y_n = a \mid X_1 = j\} \\ &= 0. \end{aligned}$$

So, the Markov chain resulting from using decision rule d will never reach state $s \in S \setminus S^*$. ■

As a last proposition, it is shown that if a decision rule is calculated from x_d according to Definition 2.3.29, decision rule d is preserved in all states $s \in S^*$.

Proposition 2.3.33:

Let $d \in D^{MD}$ be a decision rule and x_d derived from that decision rule after Equation 2.14. Furthermore, let d_x be the decision rule calculated from x_d according to Definition 2.3.29. Then

$$\mathbb{P}\{d_x(s) = a\} = \mathbb{P}\{d(s) = a\} \quad \forall a \in A_s, \quad \forall s \in S^* \setminus \tilde{S}.$$

PROOF. Suppose d , x_d and d_x as described in the proposition. Then, for $s \in S^* \setminus \tilde{S}$

$$\begin{aligned} \mathbb{P}\{d_x(s) = a\} &= \frac{x_d(s, a)}{x_d(s)} \\ &= \frac{\sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \mathbb{P}^{d^\infty}\{X_n = s, Y_n = a | X_1 = j\}}{\sum_{j \in S} \omega(j) \sum_{n=1}^{\infty} \mathbb{P}^{d^\infty}\{X_n = s | X_1 = j\}} \\ &= \frac{\sum_{n=1}^{\infty} \sum_{j \in S} \omega(j) \cdot \mathbb{P}^{d^\infty}\{X_n = s, Y_n = a | X_1 = j\}}{\sum_{n=1}^{\infty} \sum_{j \in S} \omega(j) \cdot \mathbb{P}^{d^\infty}\{X_n = s | X_1 = j\}} \\ &= \frac{\sum_{n=1}^{\infty} \mathbb{P}^{d^\infty}\{X_n = s, Y_n = a\}}{\sum_{n=1}^{\infty} \mathbb{P}^{d^\infty}\{X_n = s\}} \\ &= \frac{\sum_{n=1}^{\infty} \mathbb{P}^{d^\infty}\{Y_n = a | X_n = s\} \cdot \mathbb{P}^{d^\infty}\{X_n = s\}}{\sum_{n=1}^{\infty} \mathbb{P}^{d^\infty}\{X_n = s\}} \\ &= \frac{\mathbb{P}\{d(s) = a\} \cdot \sum_{n=1}^{\infty} \mathbb{P}^{d^\infty}\{X_n = s\}}{\sum_{n=1}^{\infty} \mathbb{P}^{d^\infty}\{X_n = s\}} \\ &= \mathbb{P}\{d(s) = a\} \quad \forall a \in A_s, \end{aligned}$$

which proves the proposition. ■

When regarding optimal basic solutions of the dual LP of POSB models, one can prove that there exists a decision rule with $x(s, a) > 0$ for at most one action a in each state s , which is useful for determining an optimal stationary deterministic policy. Theorem 2.3.34 shows a result of (Puterman, 2005), who showed the existence of an optimal basic solution and a construction of an optimal deterministic decision rule from that basic solution.

Theorem 2.3.34 (Puterman (2005), Thm. 7.2.18):

Suppose a POSB MDP with finite state and action spaces.

1. Then there exists an optimal basic solution x^* to the dual LP with a finite objective value.
2. Let $x(s, a)$ be a basic solution of the dual LP. Then for each $s \in S$, $x(s, a) > 0$ for at most one $a \in A_s$.
3. Define

$$d(s) := \begin{cases} a & \text{if } x^*(s, a) > 0 \text{ and } s \in S^* \\ \text{arbitrary} & \text{if } s \in S \setminus S^* \end{cases}$$

with $S^* := \{s \in S : \sum_{a \in A_s} x(s, a) > 0\}$, then d^∞ is optimal.

The principle ideas for the proof of Theorem 2.3.34 can be summarized as follows: The proof of the first statement is based on the assumptions of a POSB MDP from which a finite optimal value of the primal LP can be concluded. By using duality theory and the existence of a feasible solution ($x(s, a) = 0, \forall a \in A_s, \forall s \in S$) for the dual LP, the statement follows.

The second statement is proved by exploiting the structure of a basic solution. First, the problem is augmented by slack variables. Since there exist $|S|$ many constraints in the dual LP and the right hand side $\omega(s)$ are strictly greater than zero for each $s \in S$, it can be concluded that either the slack variable or at most one $x(s, a)$ per constraints can be greater than zero.

From the second statement follows that the decision rule defined in the third statement is well-defined. It is shown that the objective function of the optimal basic solution x^* is less or equal than the ω -weighted sum of the value of d^∞ . Then by using duality theory the equality $\omega^T v^* = \omega^T v^{d^\infty}$ can be followed and the optimality of d^∞ is shown.

Finally, a few more general remarks on the linear programming formulation for MDPs under the total expected reward criterion:

When using the simplex algorithm on the dual LP, this corresponds to the policy iteration algorithm where in each iteration only an action a that gives the maximum improvement over all states is updated. Or vice versa, the policy iteration algorithm equals a simplex algorithm where one action per state is pivoted.

In contrast to POSB models, it is not possible to define a general linear program for negative models. As mentioned above, the value of a negative model is the maximal solution that satisfies $v \leq \mathcal{B}v$. As a consequence, in a LP formulation, we would have to determine the maximal solution of $v \in V^-$ over

$$v(s) \leq \max_{a \in A_s} \left\{ r(s, a) + \sum_{j \in S} p(j|s, a)v(j) \right\} \quad \forall s \in S. \quad (2.16)$$

Example 2.7 is from Problem 7.22 of Puterman (2005) and presents a negative model for which the feasible regions defined through these inequalities are not convex. Therefore, the linear programming theory, that is based on polyhedrons, can not be applied.

Example 2.7:

Figure 2.6 shows a NEG MDP. The inequalities of a potential LP formulation according to Equations 2.16 for the presented MDP are:

$$\begin{aligned} v(s_1) &\leq \max\{-1 + v(s_2), -3 + v(s_3)\} \\ v(s_2) &\leq \max\{v(s_1), -1 + v(s_3)\} \\ v(s_3) &\leq v(s_3). \end{aligned} \quad *$$

Figure 2.7 shows the feasible region for $v(s_3) = 0$ which is obviously not convex.

2.4 Graph Theory and Maximum Flow Problems

This section summarizes some basic definitions from graph theory, introduces a flow network and defines the maximum flow problem. We start with the definition of a directed graph:

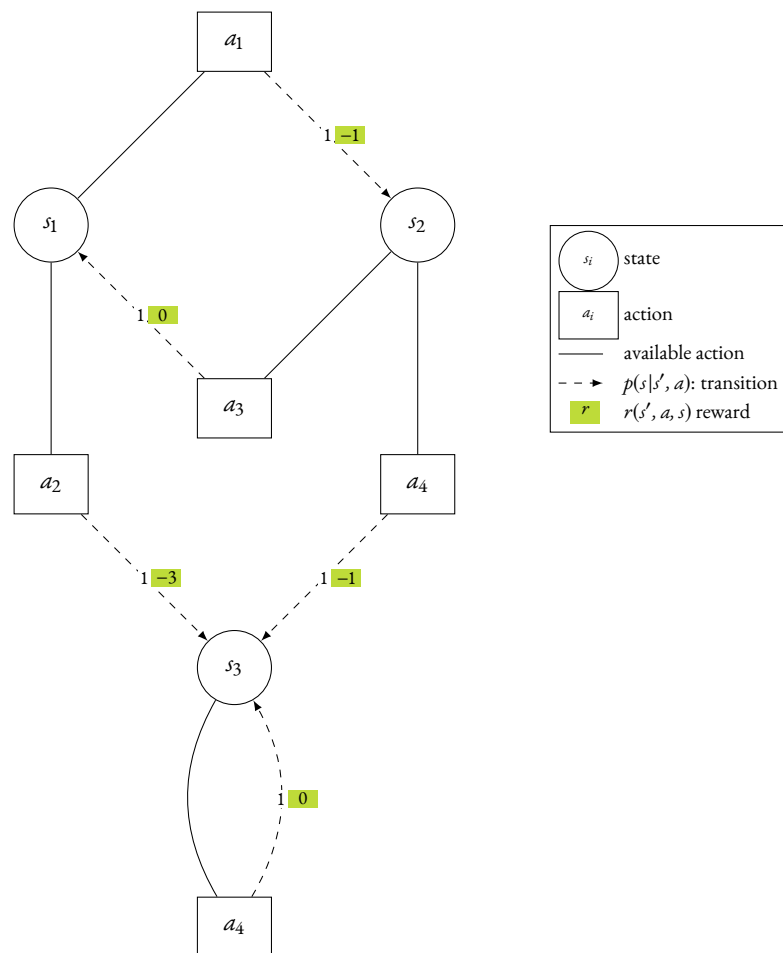


Figure 2.6: LP formulation for NEG models: non convex feasible regions

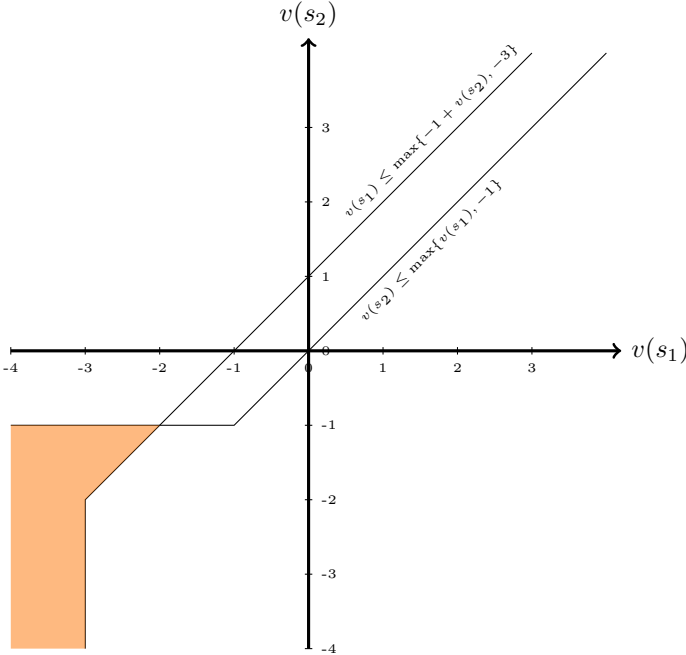


Figure 2.7: Illustration of non convex feasible regions of Example 2.7

Definition 2.4.1 (Directed Graph).

A *directed graph* $G = (N, E)$ consists of a set N of nodes and a set E of directed edges whose elements are ordered pairs of distinct nodes.

Assume for the following a directed graph $G = (N, E)$. We introduce some notation that helps to specify sets of edges or nodes in a compact way. This notation will mainly be used in Subsection 3.7 where a transformation algorithm for graphs resulting from MDPs suited for sport-strategy optimization problems is presented.

For each edge $e = (i, j) \in E$, let $start(e) = i$ be the node at which e starts and $end(e) = j$ the node at which e ends.

Let $\tilde{N} \subseteq N$ be a subset of nodes. Define $\delta_{out}(\tilde{N})$ as the set of outgoing edges of \tilde{N} , $\delta_{in}(\tilde{N})$ as the set of incoming edges to \tilde{N} and $\chi(\tilde{N})$ as the set of edges inside of \tilde{N} :

$$\begin{aligned}\delta_{out}(\tilde{N}) &:= \{e \in E \mid start(e) \in \tilde{N}, end(e) \notin \tilde{N}\} \\ \delta_{in}(\tilde{N}) &:= \{e \in E \mid end(e) \in \tilde{N}, start(e) \notin \tilde{N}\} \\ \chi(\tilde{N}) &:= \{e \in E \mid start(e) \in \tilde{N}, end(e) \in \tilde{N}\}.\end{aligned}$$

We briefly summarize some elementary objects in a directed graph.

Definition 2.4.2 (Walk).

A *walk* from $u \in N$ to $v \in N$ of length k is a sequence of edges $\omega_{u,v} = (e_1, \dots, e_k) \in \times E^k$ of the graph (N, E) that connects node u with node v , i.e., $start(e_1) = u$, $end(e_k) = v$ and $start(e_i) = end(e_{i-1})$ for all $2 \leq i \leq k$.

We defined a walk as a sequence of edges. Sometimes, also the set of nodes occurring in a walk is needed. Let $\nu((e_1, e_2, \dots, e_k))$ be the set of nodes $n_i \in N$ that lie in a sequence (e_1, e_2, \dots, e_k) of edges, i.e.,

$$\nu((e_1, e_2, \dots, e_k)) := \{n \in N \mid \exists e \in (e_1, e_2, \dots, e_k) \text{ with } start(e) = n \vee end(e) = n\}.$$

Definition 2.4.3 (Path).

A *path* $\xi_{u,v}$ from u to v is a walk $\omega_{u,v}$ without any repetition of nodes, i.e., if $|\nu(\xi_{u,v})| = |\xi_{u,v}| + 1$.

As a shorthand, $v_1 - v_2 - \dots - v_k$ will sometimes be used to denote the path $((v_1, v_2), \dots, (v_{k-1}, v_k))$, $v_j \in N$, $\forall j = 1, \dots, k$.

A path $\xi_{u,v}$ together with an edge (v, u) defines a circle. Formally, a circle is defined as a sequence of nodes:

Definition 2.4.4 (Circle).

A circle $C \in N^k$ of length k is a sequence of nodes $C = (n_1, \dots, n_k)$, where (n_{i-1}, n_i) is an edge in E for all $2 \leq i \leq k$ and all nodes are distinct except $n_k = n_1$.

To get the set of edges in a circle, we define

$$\varrho_C := \{e \in E \mid e = (n_i, n_{i+1}), n_i, n_{i+1} \text{ are subsequent nodes in } C\}.$$

Observe that ϱ_C is no path since it is a set and no tuple. Furthermore, it contains a repetitive node $n_1 = n_k$.

Finally, we specify some basic graph properties.

Definition 2.4.5 (Acyclic Graph).

A directed graph $G = (N, E)$ is *acyclic* if it contains no circle.

Definition 2.4.6 (Bipartite Graph).

A directed graph $G = (N, E)$ is *bipartite* if the node set N can be partitioned in two sets N_1 and N_2 such that $N = N_1 \cup N_2$ and for each $e \in E$ either $start(e) \in N_1 \wedge end(e) \in N_2$ or $start(e) \in N_2 \wedge end(e) \in N_1$ holds.

In Subsection 2.3.7, a linear programming formulation is deduced for sport-strategy optimization MDPs. Thereby, a flow network corresponding to the considered MDP is used to formulate a maximum flow problem.

Definition 2.4.7 (Flow Network).

A *directed flow network* is a directed graph $G = (N, E)$ where the nodes or edges have associated numerical values like, e.g., capacities or costs.

In the maximum flow problem, we wish to find the maximum flow from a *source node* $s \in N$ to a *sink node* $t \in N$. Ahuja, Magnanti, and Orlin consider a non-negative capacitated network and defines the maximum flow problem as:

Definition 2.4.8 (Maximum Flow Problem (Ahuja, Magnanti, and Orlin, 1993)).

Let $G = (N, E)$ be a capacitated network with a non-negative capacity $u_{i,j}$ associated with each edge $(i, j) \in E$. Let $s \in N$ be a source node and $t \in N$ be a sink node. Then, the maximum flow problem can be stated as

$$\left| \begin{array}{l} \max v \\ \sum_{j:(i,j) \in E} x_{i,j} - \sum_{j:(j,i) \in E} x_{j,i} = \begin{cases} v & \text{for } i = s \\ 0 & \text{for all } i \in N \setminus \{s, t\} \\ -v & \text{for } i = t \end{cases} \\ 0 \leq x_{i,j} \leq u_{i,j}, \quad \forall (i, j) \in E. \end{array} \right| \quad (\text{max flow})$$

The variables x are considered as *flow* variables and the scalar variable v as the *value* of the flow.

2.5 Relation to Markov Games

A Markov Game (MG) is a stochastic game in an MDP-like environment. Stochastic games were introduced by Shapley (1953) as a play that is controlled by two players and moves from position to position according to transition probabilities. Instead of one decision maker, there exists a whole set of players. At each decision epoch, each player chooses an action from his action set. The transition probabilities and rewards incorporate the decisions of all players. Definition 2.5.1 presents a general definition of discrete-time stationary Markov games.

Definition 2.5.1 (Markov Game).

A discrete-time stationary Markov game is a collection of objects

$$(T, S, I, \mathcal{A}_s^i, p(\cdot|s, a_1, \dots, a_k), r^i(s, a_1, \dots, a_k))$$

with the following meaning:

- $T = \{1, 2, \dots, N\}$, $N \leq \infty$ is the set of decision epochs.
- S is the set of possible system states with a single-decision game defined for each state.
- I is the set of players with $|I| = k < \infty$.
- \mathcal{A}_s^i is the set of all actions of player i at state s .
- $p(\cdot|s, a_1, \dots, a_k)$ is the transition probability function depending on the current state and the players' choices of actions.
- $r^i(s, a_1, \dots, a_k)$ is the expected reward for player i in state s given the players' choices of actions a_1, \dots, a_k ; $r_N^i(s)$ is the terminal reward for player i when the process ends in state s at decision epoch N .

An MG gets complex by the potentially different optimality criteria of the players. Assume that each player tries to maximize his expected sum of total rewards. Let π^i be the policy of player i . In the context of MGs, let π be the tuple of all player-policies, i.e., $\pi = \times_{i=1, \dots, k} \pi^i$. Then,

$$v_N^{i, \pi}(s) = \mathbb{E}_s^{\pi} \left\{ \sum_{t=1}^{N-1} r^i(X_t, Y_{1t}, \dots, Y_{kt}) + r_N^i(X_N) \right\}$$

is the value function of player i . The random process $\{X_t, Y_{1t}, \dots, Y_{kt}\}_{t \in T}$ is a Markov chain if the used policies are Markov policies.

Policies that are simultaneous best responses are in the focus of interest. Definition 2.5.2 characterizes a Nash Equilibrium in an MG with k players. The notation $\pi = (\pi^i, \pi^{-i})$ is used to distinguish between the policy π^i of player i and the policy π^{-i} of all other players.

Definition 2.5.2 (Nash Equilibrium).

A tuple of strategies $\pi^* = (\pi^{1*}, \dots, \pi^{k*})$ is a Nash equilibrium if

$$\pi^{i*} \in \arg \max_{\pi^i \in \Pi^i} v_N^{i, (\pi^i, \pi^{-i})}(s) \quad (2.17)$$

for all players $i \in I$ and all states $s \in S$.

Depending on the current state of the system, there exist different reward streams and transition functions for each player. If we assume that there exists only one state in the MG, so $|S| = 1$, we get a simultaneous move game (Anderson et al., 2007, p. 9) which is played at every decision epoch. In a setting with a finite number of states and actions, a simultaneous move game can be specified by explicitly listing all possible strategies and value functions of all players in a matrix. Furthermore, Nash (1951) proved that a simultaneous move game with a finite set of players and a finite set of actions has a Nash equilibrium of randomized policies.

The problem of finding a Nash equilibrium is PPAD-complete even for two-player games in standard form (Anderson et al., 2007, p. 16). PPAD means *Polynomial Parity Arguments on Directed graphs* and was introduced by Papadimitriou (1994). PPAD is a subclass of NP. NP-completeness is an inappropriate tool for the problem of finding a Nash-equilibrium since Nash equilibria always exist.

In the following, we focus on the simultaneous move game that occurs in a single decision epoch of an MG. For a given state, the simultaneous move game can be represented in *normal form* by a pay-off matrix. A two-person zero-sum game is a special two-person simultaneous move game, where the win of one player is the loss of the other player. Since the rewards of one player may be negative whereas the reward of an other player may be positive, we will use the term payoff instead of reward or costs in this context. So, the sum of the payoffs in a two-person zero-sum game is zero for any choices of strategies.

Definition 2.5.3 (Two-Person Zero-Sum Game).

Let $M \in \mathbb{R}^{m_i \times m_j}$ be the payoff matrix. The strategies of player i correspond to the rows of M and the strategies of player j to the columns of M . An entry M_{ij} specifies the amount that the column player j must pay to the row player i .

Since in a simultaneous move game the state is fixed, we can write a policy π of player i as a vector of \mathbb{R}^m where the j -th entry is the probability that player i chooses action i . When using strategies π^i and π^j , the expected payoff paid by the column player to the row player is

$$\pi^{iT} M \pi^j.$$

Von Neumann (1928) showed that for two-person zero-sum games the value v^* of the game equals

$$v^* = \max_{\pi^i \in \Pi^i} \min_{\pi^j \in \Pi^j} \pi^{iT} M \pi^j = \min_{\pi^j \in \Pi^j} \max_{\pi^i \in \Pi^i} \pi^{iT} M \pi^j.$$

If policy π^i of the row player is known, then $(\pi^{iT} M) \pi^j$ is a linear function in π^j . An optimal strategy of the column player is to select the column with the minimum entry of $\pi^{iT} M$. Loomis's Lemma (Borodin and El-Yaniv, 2005, p.113, Lemma 8.2) establishes that observation.

Lemma 2.5.4 (Loomi's Lemma (Borodin and El-Yaniv, 2005)):

Let

$$v_r = \max_{\pi^i \in \Pi^i} \min_j \pi^{iT} M e_j \text{ and } v_c = \min_{\pi^j \in \Pi^j} \max_i e_i^T M \pi^j.$$

Then

$$v_r = v_c = v^*.$$

PROOF. Only the proof for v_r is presented. The proof for v_c can be done analogously. By definition of v_r , we get $v_r \geq v^*$:

$$v_r = \max_{\pi^i \in \Pi^i} \min_j \pi^{iT} M e_j \geq \max_{\pi^i \in \Pi^i} \min_{\pi^j \in \Pi^j} \pi^{iT} M \pi^j = v^*.$$

Let π^{i*} and π^{j*} be the optimal strategies of both players such that $v^* = \pi^{i*T} M \pi^{j*}$ and such that Nash equilibrium strategies exist (Nash, 1951). Assume $v_r > v^*$, then $\pi^{i*T} M e_j > v^*$ for all j and we could have

$$v^* = \sum_{k=1}^{m_j} \left(\pi^{i*T} M e_j \right) \pi^{j*k} > \sum_{k=1}^{m_j} v^* \pi^{j*k} = v^*$$

which is a contradiction, hence $v_r = v^*$. ■

Loomi's Lemma makes it possible to compute the value of a two-person zero-sum game by a linear program, see 2.18 and 2.19. Observe, that without Loomi's Lemma, we would have had to concern infinitely many constraints to assure that v in 2.18 is smaller than all randomized strategies of the column player respectively that v in 2.19 is larger than all randomized strategies of the row player.

$$\left| \begin{array}{l} v_r = \max v \\ (\pi^{iT} M)_l \geq v, \forall l \in \{1, \dots, m_j\} \\ \sum_{k=1}^{m_i} \pi_k^i = 1 \\ \pi^i \geq 0 \end{array} \right| \quad (2.18)$$

$$\left| \begin{array}{l} v_c = \min v \\ (M \pi^j)_k \leq v, \forall k \in \{1, \dots, m_i\} \\ \sum_{l=1}^{m_j} \pi_l^j = 1 \\ \pi^j \geq 0 \end{array} \right| \quad (2.19)$$

The linear program 2.18 can also be found in (Anderson et al., 2007) and is exactly the dual of 2.19. From duality theory of linear programs, we know that both programs have an optimal solution with the same optimal value if and only if there exists a feasible solution for each program. It can be seen that both linear programs are feasible (just choose an arbitrary deterministic policy and v as the minimal respectively maximum entry of M). So, the existence of mixed Nash-equilibrium strategies also follows from linear programming duality theory.

Indeed an equivalence between linear programming and two-person zero-sum games was shown in Dantzig (1951) and Adler (2010). Adler completed the reduction of linear programs to two-person zero-sum games for the case that was left open by Dantzig.

Now, we go back from simultaneous move games to Markov games. Littman (1994) defines a specialization of MGs, called *two-player zero-sum Markov game*. It is a combination of a two-player Markov game with a zero-sum game in that sense that at each state $s \in S$ a zero-sum game occurs. Sport-Strategy-Optimization Markov games, defined in Subsection 3.8.2, are special two-player zero-sum Markov games.

Chapter 3

MDPs for Sport-Strategy Optimization

3.1 Introduction to MDPs in Sports Games

This section summarizes publications that handle sport-related questions by Markov processes. Since Markov chains, Markov decision processes and Markov games are closely related, approaches that use any of these frameworks are of interest. The underlying structure of all three frameworks is a stochastic process that satisfies the Markov property. The Markov property states that the next state may only depend on the current state and the chosen action, and not on the complete history of realized states.

The advantage of modeling a sports game by a stochastic process is that the evolution of the system is traceable. In contrast, a descriptive statistical analysis might determine factors that are correlated to a particular outcome. But, how these factors are influenceable may not be clear. An exaggerated example that illustrates this point: Assume, you are searching for a strategic improvement for your team in soccer. You may find by a statistical analysis that the number of penalty shots for your team is highly correlated with the number of scored goals. So, your conclusion might be that increasing the number of penalty shots for your team would be a strategic improvement. But unfortunately, a match starts with a kick-off, and it is not clear how to increase the number of penalty shots when starting from a kick-off. Of course, this example is not realistic, and a reviewer might answer that one could also find a correlation between an action, e.g., an important passing action, and the number of scored goals. Clearly, this action is influenceable by the players. However, if you observe at the match-day a different line-up of your opponent, you may be insecure whether the identified passing action is still valuable. You are not sure what mechanisms influenced your statistical results. The new line-up might have no effect up to a significant leverage effect on the value of the passing. With a dynamic model like a Markov process that accurately captures the game mechanisms, it is possible to analyze different circumstances (like a different line-up, varying day performances, etc.) with the same model.

In this thesis, a sport-strategic question is modeled from the view of a player or a team participating in a specified match. There are only a few Markov approaches that tackle strategic questions that are related to a team or a player and can be evaluated prior to a particular match. Often there is not enough data to create a model tailored to the teams participating in an upcoming game. In most of the cases where a Markov process models a sports-related question, it is an investigation of a general rule or principle.

In the following, an overview over literature that uses Markov processes to model a sport-related

issue is presented. The literature is sorted according to whether a Markov chain (MC), a Markov decision process (MDP) or a Markov game (MG) is employed. First, a summary of publications that use a Markov chain (MC) to model a sports game is provided. Since there are some works that employ a Markov chain, only works that consider volleyball or beach volleyball are presented in detail.

Florence et al. (2008) model the sequence of events occurring during a ball possession phase in women's volleyball as an MC. The data is recorded for the entire 2006 home volleyball season of the Brigham Young University Women's Volleyball Team. For each skill (pass, set, attack, etc.) the probability of an outcome (point for Brigham, point for the opponent, continuation of the rally) is evaluated by calculating the unconditional probability for that outcome from the transition probability matrix. The distribution for each unconditional probability is calculated using Gibbs sampling and determines the variability of the unconditional probability point estimates. The implications from that investigation are general recommendations for Brigham Young University Women's Volleyball Team like: "If the high and inside delivery can be avoided the attack has a good probability of being successful." (Florence et al., 2008, p. 14)

Similar Miskin, Fellingham, and Florence (2010) investigate skill importance in women's volleyball. The authors model play sequences as discrete absorbing MCs by using a Bayesian approach to estimate the transition probabilities from the data gathered. The data was collected during the 2006 competitive season of a single women's Division I volleyball team of the National Collegiate Athletic Association in the United States. The 36 states consolidated in this analysis are moves that consist of a skill and a rating combination, e.g., a set is rated according to its distance from the net. The importance score of a skill is a metric that incorporates its impact on the desired outcome and its uncertainty. It is computed by the posterior distribution associated with the skill.

Ferrante and Fonseca (2014) use an MC approach for volleyball to compute an explicit formula for the serving team's winning probability in a set. Besides this, the mean duration of a set is computed regarding the expected number of rallies. The authors assume that the probability of winning a single rally is independent of the other rallies and constant during the game. The states in their model correspond to different scores that may occur in a set together with an indicator which team serves next. The winning probability is computed concerning two parameters which represent the likelihood of winning a rally depending on the serving team. MC properties and combinatorial arguments are used to derive the explicit formula for the winning probability.

Besides volleyball, there are other works using MC-approaches in different sports like, e.g., Newton and Aslam (2009) in tennis, Heiner, Fellingham, and Thomas (2014) and Liu and Hohmann (2013) in soccer, Pfeiffer, Zhang, and Hohmann (2010) in elite table tennis, Bukiet, Harold, and Palacios (1997) in baseball, Shirley (2007) and Štrumbelj and Vračar (2012) in basketball, and McGarry and Franks (1994) in squash.

Next, works are presented that employ Markov decision processes (MDPs) for tackling a sports-related issue. The literature is sorted by the sport studied:

Tennis:

Clarke and Norman (2012) as well as Nadimpalli and Hasenbein (2013) investigate an MDP for tennis games to determine when a player should challenge a line call. The latter one is the more detailed model. It is described briefly in the following: A decision point occurs when an opportunity to challenge the umpire arises. The states include the outcome of the point, the score, the number of challenges remaining, the probability that the call is incorrect, and the result of a successful challenge. There are

two possible actions in each state: *challenge* and *do not challenge*. Further parameters of the model are the relative strength of the players and the fallibility of the officials. These parameters are used to generate the transition probabilities for the model. They use the standard linear programming approach for multi-chain, average cost MDPs to obtain optimal policies under a variety of parameter settings.

Chan and Singal (2016) use an MDP to compute an optimization-based handicap system for tennis. The weaker player gets 'free points' at the start of the match, such that the match-win probability of both players is equalized. The input of the model is the point-win probability of each player. A standard policy iteration solves the model. The resulting optimal policy specifies at which point the weaker player should use a free point.

Norman (1985) builds an aggregated MDP for tennis games to tackle the question when to serve fast or when to serve slow at each stage of a game. The model is solved analytically using a monotonicity property of the optimal cost function and dynamic programming. The optimal strategy specifies, depending on the point-winning probability for fast and slow serves, which kind of service should be used throughout the match.

Terroba et al. (2013) develop a more detailed MDP-based framework for tennis matches. The information needed to build the model is semi-automatically gathered from broadcast sports videos. Machine learning algorithms are executed to identify optimal policies due to the large state space ($\approx 10^5$ state-action pairs). A Monte Carlo tree search algorithm is applied to estimate the value function. Famous tennis matches of the past are investigated in experiments (the 2010 Australian Open Women's Semi Final between Na Li and Serena Williams as well as the 2009 French Open fourth Round match between Rafael Nadal and Robin Söderling). The results of the Monte Carlo tree search algorithm are state-action pairs with a high estimated value function. They present how the player who has lost in reality could have won the match with identical skills, just by using a different policy.

Soccer:

Hirotsu and Wright model soccer as a four-state Markov Process. They make general tactical consideration as in some cases "[...] the players do not have time to pause and consider rationally what exactly to do next. This means that tactical considerations are most valuable when they can be expressed in terms of general principles, for the benefit of coaches as much as players." (Wright and Hirotsu, 2003, p. 1). Their tactical considerations concern the optimal timing of a substitution (Hirotsu and Wright, 2002), the best strategy for changing the configuration of a team (Hirotsu and Wright, 2003b), and to determine under which circumstances a team may benefit from a professional foul (Wright and Hirotsu, 2003).

Cricket:

Clarke and Norman (1998) formulate an MDP for cricket to determine whether the batsman should take an offered run when maximizing the probability that the better batsman is on strike at the start of the next over. The model is solved analytically by dynamic programming. The optimal policy takes the run or not depending on the relation between the scoring probabilities of the good and the bad batsman. A similar analysis with a different objective is done in Clarke and Norman (1999).

Baseball:

Hirotsu and Wright (2003a) formulate a Markov model for baseball to calculate an optimal pinch-hitting strategy under the 'Designated Hitter Rule'. Their method can be applied to a specific match by using the probability of each player to achieve a single, double, triple, home run, walk or out.

Most of the presented models (all except the detailed model of Terroba et al. (2013)) require as an

input a point-winning probability or scoring probability. These aggregated transition probabilities include the opponent's skills and strategy. Therefore, it may be hard to apply those models to a particular match for instance if the teams have not recently played against each other. In contrast, the larger models (like the model of Terroba et al. (2013)) which require only transition probabilities that capture smaller events are not solvable by classical algorithms for MDPs. Instead, some local search or approximation algorithm is used to identify valuable actions.

To the best of our knowledge, there exist only a few publications that model a sport-strategic question by a Markov game (MG). This may be the case because determining a Nash-Equilibrium is in general PPAD-complete (Anderson et al., 2007).

However, baseball is a special case. The sequential and discrete nature of the sports makes it possible to use dynamic programming for determining a Nash-equilibrium. For instance, Kira et al. (2015) formulate an MG for baseball and compute Markov perfect equilibria. The transition probabilities of the MG are assumed to depend only on the probability parameters for the hitting skills of the players. They use a dynamic programming algorithm for solving the Bellman equations that characterize the value function of the game for both teams. This approach is possible since the actions are chosen sequentially and not simultaneously.

Again, the decision question often concerns a general rule or principle, and the input probabilities contain large parts of the game mechanisms. For instance, Turocy (2008) uses MGs fed with a lot of historical data to clarify whether there *has been* a "last-up" advantage in baseball *on average in the past*, or Anbarci, Sun, and Ünver (2015) try to decide on the fairness of tie-break mechanisms in soccer on the basis of MG models.

Sarkar (2018) is another author who examines a general principle. He tries to find evidence for the inverse relation between crosses and goals in soccer. A simultaneous move game with two defending strategies (high defensive line, low defense line) and two attacking strategies (cross, short through pass) is constructed. The payoffs are winning probabilities multiplied with the magnitude of gain respectively loss. The winning probabilities are calculated from very rough probabilities like the probability of breaking the offside trap or the goalkeeper's save rate. A mixed Nash equilibrium is calculated that suggest that teams with a greater chance of scoring from crosses use the crosses less frequently. The reason for it is that the defending team uses an offside trap more regularly for better teams.

Routley and Schulte (2015) employ MGs to rank ice-hockey players according to their skills. An upgraded MG of (Schulte et al., 2017) also includes location information.

Walker, Wooders, and Amir (2011) use binary Markov games to model sports games like tennis. They show that under specific monotonicity properties optimal policies to win the match are a repeated application of an optimal policy to win a rally. This finding fits to the analytical result found for the rougher SSO-MDP presented in the next chapter.

From the described literature, it can be seen that there exist some works that consider a Markov process as a basis for modeling a sports game. Regardless of whether an MC, an MDP or an MG is used, all works face the same conflict: Either a manageable model with rough transitions is built that can be solved, or a model with more detailed transitions is developed, but those models get so large such that only an approximate solution can be found. The small, manageable models mostly require transition probabilities that include events of both teams participating in the match. Those are hard to estimate for a particular pairing of teams. Therefore, those models are often used to examine general questions or principles such that an optimal policy depending on the transition probabilities can be computed.

However, how these probabilities can be estimated for a match, which has not yet taken place, stays often unclear.

3.2 Definition of Sport-Strategy Optimization MDPs (SSO-MDPs)

In this section, a general class of MDPs suitable for modeling sports games and answering strategic questions is specified. Before defining a special MDP class, it is motivated why Markov decision processes and not Markov games are used to model a sport-strategic question. A formalization of a sport-strategic question is given in Definition 5.1.1 in Chapter 5. For the moment, a sport-strategic question can be considered as a question that asks for the best playing strategy in a particular match against a particular opponent team.

Since this thesis focuses on sports games, which are often team sports, the term “team” will be used throughout this chapter. However, all considerations are also applicable for sports games where teams consist only of one player – like e.g. tennis. Furthermore, in a sport-related context, the term “strategy” is used if formally a decision rule of a stationary policy is meant.

The solution of a Markov game (MG) is a Nash equilibrium, which consists of simultaneous best strategies of both teams. This means, the optimal strategy of a team in an MG is the best response to a strategically perfect playing opponent. In sports games, the participants of a match behave according to practiced playing patterns and try to perform best when sticking to their strategic plan. By “performing best” it is meant that each player tries to carry out an action as best as possible in a given situation. For example in soccer, a pass that is part of a playing sequence and can be performed more or less precisely. Of course, each player tries to pass as accurately as possible. But, no player would suddenly deviate from the playing pattern and pass to another player if that is totally unpredictable for his teammates. Especially in team sports, where a coordination between the players is necessary, a player would rather stick to the playing pattern instead of doing an action that might be optimal but unpredictable for his teammates.

Also in an MDP, it is possible to model an opponent team completely analogous to the team whose strategy should be optimized. But instead of solving a minimax problem, the opponent plays a fixed strategy. By fixing the opponent’s strategy, the opponent team becomes a part of the environment and can be captured in the transition probabilities. But still, it is possible to analyze optimal strategies against different opponent strategies. Solving several MDPs with respect to different opponent strategies leads to a Markov game with a discrete set of strategies.

In all sports games, the objective of each team is to win the match. Winning or losing a match is determined by the rules of the respective sports game. Some sports games are won if a certain condition of the score is met, other sports games have a fixed time period after which the winner of the match is determined. In most leagues or tournaments it is only of second interest whether the match is won by a large lead or not. In first case, it matters whether the match is won, lost or, if possible, a draw occurred. This fact leads to the idea of modeling a sports game by an MDP with absorbing states which correspond to states at which the match has terminated. And furthermore, to use a reward function that returns a reward of 1 for a transition to a winning state while all other transitions are 0-reward transition. Together with the expected total reward criterion (Definition 2.1.4), the objective in such an MDP is equivalent to maximize the probability of winning the match, which will be formally shown in Subsection 3.4.

By modeling a losing state as an absorbing state and maximizing the probability of reaching a winning state, the losing states can be viewed as dead ends that can not be compensated and must be avoided in any case. This modeling decision can be justified from a sporting perspective: For example, in tournaments, e.g., in a quarterfinal of a world championship, a loss of the match and therefore an “out” of the tournament cannot be compensated. Also in a regular season, losing several matches, but having, e.g., the best passing performance, can not balance out the lost points in the league table at the end of the season. This reasoning suggests that it is not appropriate to model a losing state with some finite penalty that could be compensated by some “well performed” actions that may have a small positive reward. Instead, as described in the last paragraph, losing states should be modeled as dead ends that must be avoided in any case. This is modeled by the objective that maximizes the probability of ending in a winning state.

Since the strategy of a team which is participating in the match should be optimized, the decisions in the MDP will correspond in some way to action choices of the team. Without specifying the definition of a team action further, one can easily think of sports games where the number of team actions – and therefore the number of decision points – is not known a priori the match. For example, in beach volleyball a team action could be modeled as a field attack of a team. The number of field attacks a team performs in a rally or a set is not known and differs between matches. An MDP with absorbing states is an adequate tool to model an indefinite-horizon MDP. As the system reaches a goal state, no further rewards or costs are accumulated. Furthermore, it is not predetermined after how many steps the system reaches an absorbing state.

In a sports game, the set of possible starting states of a match or set is known. For example, in soccer, a match starts with a kick-off; or in tennis a match starts with a serve. Even if only a set of possible starting states is known, an artificial starting state can be introduced that has transitions to all possible starting states. The transition probability from the artificial starting state to each possible starting state may equal a meaningful distribution of the initial state. Therefore, it is no restriction to assume a single starting state s_1 for SSO-MDPs. The knowledge of a starting state is crucial when applying heuristic methods. Some states may become irrelevant when starting from a certain initial state and an investigation of the entire state space may be avoidable.

Finally, a crucial assumption that characterizes sports games is made: In a sports game, there will always exist a maybe small but strictly positive probability that the match is lost even if the team is dominating the opponent team and plays the strategical optimal strategy. As a support of this hypothesis, OddsShark presents statistics about the win percentage of underdogs. These are especially in the Major League Baseball and the National Hockey League very high (above 40%). Furthermore, the property that no team can win for sure is a fact that may explain why sports games are so popular. Even if your preferred team is the complete outsider in a match, there exists a, maybe little but positive, chance that your team will win the match.

Let X_t be a random variable that captures the state of the process at time t . Furthermore, let L be the set of losing states and W be the set of winning states. The described assumption can be formalized as follows:

Assumption 3.2.1 (No policy guarantees winning):

For all states $s \in S \setminus (L \cup W)$ and every policy $\pi \in \Pi$, there exists a strictly positive probability of losing the game, i.e.,

$$\mathbb{P}_L^\pi(s) := \sum_{t=1}^{\infty} \mathbb{P}^\pi \{X_t \in L, X_{t'} \notin L \cup W, \forall t' < t, X_1 = s\} > 0.$$

Finally, all properties of a sport-strategy-optimization-MDP (SSO-MDP) described above are summed up in the following definition:

Definition 3.2.2 (Sport-Strategy-Optimization-MDP).

A *sport-strategy-optimization-MDP (SSO-MDP)* is a tuple

$$(S, A, p(\cdot|s, a), r(s, a), W, L, s_1)$$

that satisfies Assumption 3.2.1 and where

- S is a finite set of n states.
- A is a finite set of m actions with $A_s \neq \emptyset \forall s \in S$.
- $p(\cdot|s, a)$ is a stationary transition probability function which satisfies

$$p(s|s, a) = 1 \quad \forall s \in W \cup L, \forall a \in A_s.$$

- $r(s, a)$ is a stationary expected reward function which satisfies

$$r(s, a, s') = \begin{cases} 1, & \forall s \in S \setminus (W \cup L), s' \in W, a \in A_s \\ 0, & \text{else.} \end{cases}$$

- $W \subset S$ is a non-empty set of winning states.
 - $L \subset S$ is a non-empty set of losing states which satisfy $L \cap W = \emptyset$.
 - $s_1 \subset S$ is a known starting state.
-

Since an SSO-MDP has by definition only finitely many states, it can be concluded from Assumption 3.2.1 that already after n many steps there must occur a positive probability for a transition to a losing state:

$$\sum_{t=1}^n \mathbb{P}^\pi \{X_t \in L, X_{t'} \notin L \cup W, \forall t' < t, X_1 = s\} > 0, \forall s \in S \setminus (L \cup W), \forall \pi \in \Pi.$$

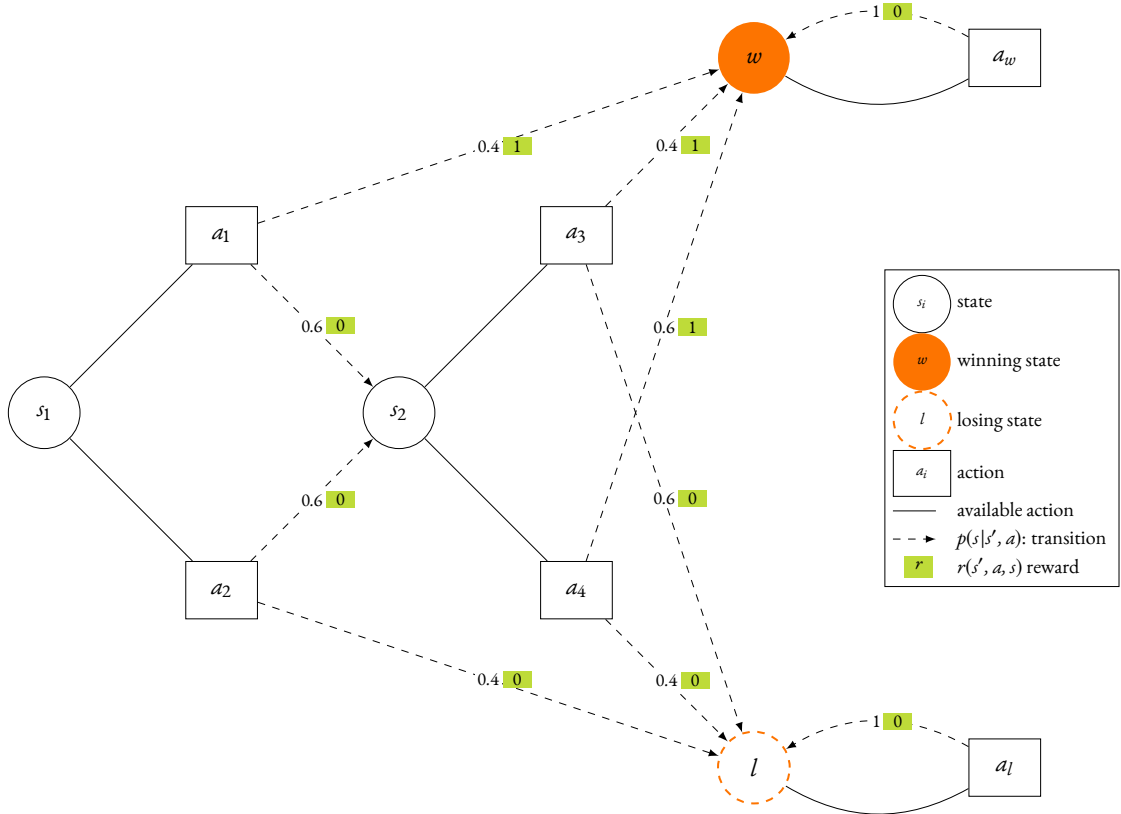


Figure 3.1: SSO-MDP Example

Figure 3.1 illustrates a graphical representation of an SSO-MDP example. A formal definition of a graph associated with an SSO-MDP is given in Subsection 3.6. In the graphical representation of an SSO-MDP, each action must have a unique predecessor state. Therefore, for general non-unique action sets \mathcal{A} an augmented action set \mathcal{A}' consisting of state-action pairs is defined in the following way:

$$\begin{aligned} \mathcal{A}' &\subseteq \mathcal{S} \times \mathcal{A}, \\ \mathcal{A}'_s &:= \{(s, a) \mid a \in \mathcal{A}_s\}. \end{aligned}$$

In the augmented action set, the state is included in the action. Thereby, an action $a \in \mathcal{A}$, which may be available in more than one state, becomes unique in \mathcal{A}' . It is possible to evaluate \mathcal{A}'^{-1} by $\mathcal{A}'^{-1}(a') = \mathcal{A}'^{-1}(s, a) = s$.

As described in the legend of Figure 3.1, the available actions in a state are connected to the state via a solid line. From each action, the outgoing dashed lines mark the possible transitions together with the transition probabilities and the rewards. The special states $w \in \mathcal{W}$ and $l \in \mathcal{L}$ have a special look which indicates that they are absorbing states.

For verifying that the MDP presented in Figure 3.1 is an SSO-MDP besides the requirements on the absorbing states and the reward function, Assumption 3.2.1 has to be checked. Due to the small problem size, it can be seen that from each action there exists a path to the losing state l state where all

transition probabilities are greater zero. So, it can be concluded that starting from all $s \in S$ and using any policy $\pi \in \Pi$ the probability of losing the game is greater zero.

The reader may ask how hard or easy it will be to check Assumption 3.2.1 for larger SSO-MDPs, where the structure of the problem can not be presented on half a page. As it will be seen later, when investigating concrete SSO-MDPs for sports games, the actions in an SSO-MDP will correspond in some degree of granularity to team actions. And at this point, it becomes important that the model captures a sports game. An action in a sports game is a physical effort which has always an opportunity to fail. The reader may remember, e.g., soccer scenes where the goalkeeper fails to control a shot that was targeted on him with a normal speed. Although those events happen only very rarely, even the best goalkeepers make such big mistakes, e.g., Karius in the Champions League final of 2018.

3.3 Classification

This sections classifies SSO-MDPs, see Definition 3.2.2, according to the SSP MDP classes presented in Subsection 2.3.1. The definition of an SSO-MDP does not explicitly specify a set of goal states G . It is not goal oriented like MDP classes directly derived from SSP MDPs. However, two sets of absorbing states, which are called winning states W and losing states L , are defined. In the following, different choices of G for SSO-MDPs are considered together with the implications that follow from these choices.

But first, it is shown that an SSO-MDP belongs to the class of POSB MDPs which is also a non-goal oriented class.

Theorem 3.3.1:
SSO-MDP \subset POSB MDP

PROOF. To show that every SSO-MDP is a POSB MDP, the two conditions of a POSB MDP, which are $v_+^\pi(s) < \infty$ for all $s \in S$ and $\pi \in \Pi$ and for each $s \in S$ there exists at least one $a \in \mathcal{A}_s$ with $r(s, a) \geq 0$, have to be verified.

Since a strictly positive reward occurs only from a transition to an absorbing winning state w , each process can only accumulate once a strictly positive reward. Therefore, the sum of positive rewards $v_+^\pi(s)$ is finite for all states s and all policies π .

In an SSO-MDP, there exist only non-negative rewards and each state s contains at least one action. So, the second assumption of a POSB MDP is satisfied for each state $s \in S$. ■

In Subsection 2.3.1, a proof of Kolobov, Mausam, and Weld is cited. They show that every POSB MDP can be converted into a GSSP MDP with identical optimal policies. In the stated conversion, all strongly connected components without outgoing edges and whose internal edges correspond only to 0-reward actions are considered as goal states of the resulting GSSP MDP. Of course, this kind of conversion is also possible for SSO-MDPs. However, the set of goal states should be specified explicitly like in the definition of an SSP MDP.

By setting $G := W$, a MAXPROB MDP is received from the SSO-MDP. But, an SSO-MDP with goal states defined like this will not be a GSSP MDP. From an intuitive view, this can be justified as

follows: A proper policy is a policy that reaches with probability 1 a goal state which would be in this setting a winning state of the sports game. This is a contradiction to Assumption 3.2.1 which states that for each policy in each state the probability of losing the game is strictly greater than zero.

Observation 3.3.2:

Assume an SSO-MDP and define $G := W$. Then an SSO-MDP with the defined set of goal states is a MAXPROB MDP but no GSSP MDP.

PROOF. An SSO-MDP with $G := W$ satisfies the definition of a MAXPROB MDP (Definition 2.3.13): The reward function fulfills the requirement of a MAXPROB MDP that every transition to a goal state yields a reward of 1 while all other transitions are 0-reward transitions.

Assume an SSO-MDP is a GSSP. Then there would exist a proper policy in s_1 , i.e., a stationary policy d that satisfies

$$\mathbb{P}^{d^\infty} \{X_n \notin G \mid X_1 = s\} < 1, \quad \forall s \in S \text{ reachable from } s_1,$$

where n is the number of states reachable from s_1 . By Assumption 3.2.1 of an SSO-MDP, a losing state $l \in L$ is reachable from s_1 under each decision rule d . However, as l is an absorbing state

$$\begin{aligned} & \mathbb{P}^{d^\infty} \{X_n \notin G \mid X_1 = l\} \\ &= \mathbb{P}^{d^\infty} \{X_n \notin W \mid X_1 = l\} \\ &\geq \mathbb{P}^{d^\infty} \{X_n \in L \mid X_1 = l\} \\ &= 1, \end{aligned}$$

which is a contradiction.

So, in SSO-MDPs there can not exist a proper policy starting in s_1 . SSO-MDPs with $G := W$ can not be GSSP MDPs. ■

In the following, an SSO-MDP is considered as an SSP MDP with goal states $G := W \cup L$. By using this definition for the set of goal states and the reward function, the SSO-MDP does not fulfill the requirements of a MAXPROB MDP:

Observation 3.3.3:

Let $G := W \cup L$. Then, SSO-MDP $\not\subseteq$ MAXPROB MDP.

PROOF. A MAXPROB MDP requires that each transition to a goal states gives a reward of 1. When considering the losing states and winning states as goal states, this requirement is not satisfied. A transition to a losing state only gives a reward of 0. ■

However, an SSO-MDP with $G := W \cup L$ is a GSSP MDP:

Theorem 3.3.4:

Let $G := W \cup L$. Then, SSO-MDP \subset GSSP MDP.

PROOF. To show that an SSO-MDP with $G := W \cup L$ is a GSSP MDP, two conditions have to be verified: There must exist a proper policy rooted at s_1 and the sum of non-negative rewards of any policy is finite in every state s reachable from s_1 . The second condition is clearly fulfilled since it was already shown in Theorem 3.3.1 that an SSO-MDP is a POSB MDP.

For proving the existence of a proper policy, it is used that each SSO-MDP satisfies Assumption 3.2.1:

$$\mathbb{P}_L^\pi(s) > 0 \quad \forall s \in S \setminus (L \cup W), \quad \forall \pi \in \Pi$$

where $\mathbb{P}_L^\pi(s)$ is defined as

$$\sum_{t=1}^{\infty} \mathbb{P}^\pi \{X_t \in L, X_{t'} \notin L \cup W, \forall t' < t, X_1 = s\}.$$

Since we face an MDP with $|S| = n$ and stationary data, for every policy $\pi \in \Pi$ and every starting state $s \in S \setminus (L \cup W)$ there exists a $t \leq n$ with

$$\mathbb{P}^\pi \{X_t \in L, X_{t'} \notin L \cup W, \forall t' < t, X_1 = s\} > 0.$$

All losing states $l \in L$ are absorbing state. If the process reaches l at time $t \leq n$, it will stay in l . The probability that after n time steps the process is in state l can be calculated by

$$\begin{aligned} & \mathbb{P}^\pi \{X_n \in L, X_1 = s \notin L \cup W\} \\ &= \sum_{t=1}^n \mathbb{P}^\pi \{X_t \in L, X_{t'} \notin L \cup W, \forall t' < t, X_1 = s\} \end{aligned}$$

and is greater zero. So, for every policy $\pi \in \Pi$ and ever state $s \in S \setminus (L \cup W)$

$$\begin{aligned} 0 &< \mathbb{P}^\pi \{X_n \in L, X_1 = s\} \\ &= 1 - \mathbb{P}^\pi \{X_n \in S \setminus L, X_1 = s\} \\ &\stackrel{W \cup L \subset S}{\leq} 1 - \mathbb{P}^\pi \{X_n \in S \setminus (L \cup W), X_1 = s\} \\ &\stackrel{G := W \cup L}{=} 1 - \mathbb{P}^\pi \{X_n \notin G, X_1 = s\}. \end{aligned}$$

The last equation can be reformulated as

$$\forall \pi \in \Pi, \forall s \in S \setminus G : \mathbb{P}^\pi \{X_n \notin G, X_1 = s\} < 1$$

which proves that every policy $\pi \in \Pi$ from every starting state $X_1 = s$ is proper. ■

Theorem 3.3.5:

Let $G := W \cup L$. Then, SSO-MDP \subset Bertsekas-SSP MDP

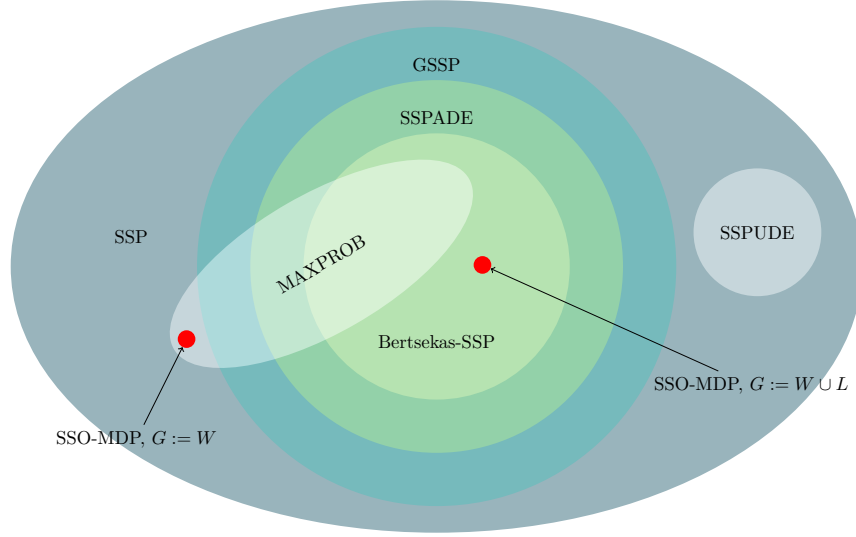


Figure 3.2: MDP classes hierarchy SSO-MDPs included

PROOF. We have shown in the proof of Theorem 3.3.4 that there exists a proper policy for each starting state $s \in S$. So the first assumption of Bertsekas-SSP MDPs is satisfied.

Since in the proof of Theorem 3.3.4 it is shown that all policies in an SSO-MDP are proper, the second condition of Bertsekas-SSP MDPs, which is *every improper policy must incur a reward of $-\infty$* , does not concern any policy. ■

Summing all results up, we get that every SSO-MDP is a POSB MDP. By setting the set of goal states to $W \cup L$ an SSO-MDP is also a Bertsekas-SSP MDP with the property that every policy is proper. In Figure 3.2, SSO-MDPs are included in the diagram of SSP classes considered in this thesis.

3.4 Theoretical Analysis

This section analyses the structure of SSO-MDPs and the properties that can be concluded from it. Those properties are of importance when an SSO-MDP is solved for analyzing a sports-related issue. This section starts with a formal proof that the objective value of a policy equals the winning probability. It goes on with convergence statements regarding policy iteration and value iteration. Finally, it is shown that the dynamic programming operator applied to SSO-MDPs is a contraction mapping. An explicit formula of the contraction factor and its interpretation in the context of SSO-MDPs is given.

Due to the special structure of the reward function in an SSO-MDP, the expected total reward of a policy π equals the probability that the system reaches a winning state.

Proposition 3.4.1 (Objective Function of an SSO-MDP):

Assume an SSO-MDP starting in a state $s \in S \setminus \{W \cup L\}$ and a policy π , then

$$v^\pi(s) = \sum_{t=2}^{\infty} \mathbb{P}^\pi \{X_{t-1} \notin W, X_t \in W \mid X_1 = s\}.$$

PROOF. The expected value of the reward process concerning policy π is defined as

$$\begin{aligned} & v^\pi(s) \\ &= \mathbb{E}_s^\pi \left\{ \sum_{t=1}^{\infty} r(X_t, Y_t) \right\} \\ &= \sum_{(s_1, a_1, s_2, \dots) \in (S \times \mathcal{A})^\infty} \left(\sum_{t=1}^{\infty} r(s_t, a_t, s_{t+1}) \right) \mathbb{P}^\pi \{ X_1 = s_1, Y_1 = a_1, X_2 = s_2, \dots \mid X_1 = s \}, \end{aligned}$$

where \mathbb{P}^π is the probability for the sample path $(s_1, a_1, s_2, \dots) \in (S \times \mathcal{A})^\infty$ under policy π . An investigation of \mathbb{P}^π can be found in Subsection 2.1.1.

Consider a sample path $(s_1, a_1, s_2, \dots) \in (S \times \mathcal{A})^\infty$. If there exists a point in time at which the realization occupies a winning state, the reward of the transition to the winning state is 1. If no such point in time exists, the sum over all rewards is 0. So, the last sum simplifies to:

$$\sum_{\substack{(s_1, a_1, s_2, \dots) \in (S \times \mathcal{A})^\infty \\ \text{with } \exists t \in \mathbb{N} : s_t \in W}} \left(\sum_{t=1}^{\infty} r(s_t, a_t, s_{t+1}) \right) \cdot \mathbb{P}^\pi \{ X_1 = s_1, Y_1 = a_1, X_2 = s_2, \dots \mid X_1 = s \}$$

Once a winning state has been entered, the random process stays in the winning state due to its absorbing property. Therefore, any realization that enters a winning states does it only once and generates a reward of 1. All sample paths that enter at time t the first time a winning state can be summed up, which is

$$\sum_{t=1}^{\infty} \sum_{\substack{(s_1, a_1, s_2, \dots) \in (S \times \mathcal{A})^\infty \\ \text{with } s_{t-1} \notin W, s_t \in W}} 1 \cdot \mathbb{P}^\pi \{ X_1 = s_1, Y_1 = a_1, X_2 = s_2, \dots, X_t = s_t \mid X_1 = s \}.$$

Since the starting state s is assumed not to be in W , the sample path starts with s and cannot be in a winning state at time $t = 1$. Therefore, the last sum equals

$$\sum_{t=2}^{\infty} \sum_{\substack{(s_1, a_1, s_2, \dots) \in (S \times \mathcal{A})^\infty \\ \text{with } s_{t-1} \notin W, s_t \in W}} \mathbb{P}^\pi \{ X_1 = s, Y_1 = a_1, X_2 = s_2, \dots, X_t = s_t \}$$

Hence, the expected total reward from using policy π equals the probability that under π the random process enters at some point in time a winning state given that the process has started in state $s \in S \setminus \{W \cup L\}$ which can also be written as

$$\begin{aligned} & \sum_{t=2}^{\infty} \mathbb{P}^\pi \{ \omega = (s, a_1, s_2, \dots) \in (S \times \mathcal{A})^\infty : X_{t-1}(\omega) \notin W, X_t(\omega) \in W \} \\ &= \sum_{t=2}^{\infty} \mathbb{P}^\pi \{ X_{t-1} \notin W, X_t \in W \mid X_1 = s \}. \end{aligned} \quad \blacksquare$$

For Bertsekas-SSP MDPs, it is known that the optimality equations have a unique fixed point. This is outlined in Subsection 2.3.4 and based on the result of Bertsekas, 2001, Prop. 2.1.2. In the last subsection, we have seen that SSO-MDPs with $G := L \cup W$ are special Bertsekas-SSP MDPs. So, this result also follows for SSO-MDPs.

Theorem 3.4.2:

The optimality equations for SSO-MDPs have a unique fixed point.

PROOF. SSO-MDPs are special Bertsekas-SSP MDPs, see Theorem 3.3.5, and optimality equations for Bertsekas-SSP MDPs have a unique fixed point (Bertsekas, 2001, Prop. 2.1.2). ■

Since the value of an MDP under the total expected reward criterion satisfies the optimality equations (Theorem 2.3.24), it can be followed from Theorem 3.4.2 that value iteration converges to an ϵ -approximation of the value of the SSO-MDP for every value function $v \in V$. Remember that this does not hold for general MDPs under the total expected reward criterion.

Corollary 3.4.3 (Value Iteration converges for SSO-MDPs):

For SSO-MDPs, value iteration converges to an ϵ -approximation of the value of the SSO-MDP for every value function $v \in V$.

For Bertsekas-SSP MDPs policy iteration generates in each step a strictly better policy (Bertsekas, 2001, Sec. 2.2). So again, this also holds for SSO-MDPs, and it can be derived that the policy iteration algorithm terminates after finitely many steps at an optimal policy.

Corollary 3.4.4 (Policy Iteration converges for SSO-MDPs):

For SSO-MDPs, policy iteration converges after finitely many iterations to an optimal policy.

In SSO-MDPs there is not only a proper policy for every start state. Even every policy is proper. This was already shown in the proof of Theorem 3.3.4 in the last subsection.

Theorem 3.4.5:

Every policy π in an SSO-MDP is proper.

PROOF. In proof of Theorem 3.3.4, we have already shown that all policies are proper policies in an SSO-MDP. ■

For MDPs under the total expected reward criterion for which all policies are proper policies, it can be shown that the dynamic programming operator is a contraction mapping.

Theorem 3.4.6 (Dynamic Programming Operator is contraction [Bertsekas (2001), p. 94]:
In SSO-MDPs, the dynamic programming operator is a contraction mapping. This means, there exists positive constants ω_i for all $i \in S$ and a constant $\gamma \in [0, 1)$ such that

$$\|\mathcal{B}v - \mathcal{B}u\|^\omega \leq \gamma \|v - u\|^\omega$$

for all value functions $v, u \in \mathbb{R}^n$.

In this notation, $\|\cdot\|^\omega$ is the L_∞ -norm where each vector is scaled by ω . This means,

$$\|u\|^\omega := \max_{i=1, \dots, n} |u_i \cdot \omega_i^{-1}|.$$

Thus, the inequality of Theorem 3.4.6 can be written as

$$\max_{i=1, \dots, n} \frac{1}{\omega_i} |(\mathcal{B}v)_i - (\mathcal{B}u)_i| \leq \max_{i=1, \dots, n} \gamma \cdot \frac{1}{\omega_i} \cdot |v_i - u_i|.$$

Tseng gives in Tseng (1990) a proof of Theorem 3.4.6 and specifies, in contrast to (Bertsekas, 2001), the contraction factor γ . In the following some steps of the proof of Tseng, 1990, Lemma 3 are reviewed to gain insights into the contraction factor of SSO-MDPs:

In SSP MDPs, where only proper policies exist, all states except the goal states G can be partitioned into non-empty subsets S_1, \dots, S_r such that for any $s \in \{1, \dots, r\}$, $i \in S_s$ and $a \in A_i$, there exists some $j' \in G \cup S_1 \cup \dots \cup S_{s-1}$ such that $p(j'|i, a) > 0$. Then, Tseng defines the weights ω as

$$\omega_i := 1 - v^{2^s}, \quad \forall i \in S_s, \quad \forall s = 1, \dots, r,$$

where $v_j := \min_{a \in A_i, i \in S} \{p(j|i, a) \mid p(j|i, a) > 0\}$. With this weights the contraction factor γ is

$$\gamma = \frac{1 - v^{2^{r-1}}}{1 - v^{2^r}}.$$

So, the contraction factor depends on the number of state subsets r and the minimal smallest transition probability v specified in the model.

In the following, it is outlined how this result can be interpreted in the context of SSO-MDPs. First, the contraction factor γ is monotone decreasing for $v \in (0, 1)$. This can be seen from the derivative

$$\frac{\partial}{\partial v} \left(\frac{1 - v^{2^{r-1}}}{1 - v^{2^r}} \right) = \frac{v^{2^{r-2}}(-v^{2^r} + 2r(v-1) + 1)}{(1 - v^{2^r})^2}.$$

For $v \in (0, 1)$, the denominator and the first factor in the numerator are always positive. The second factor in the numerator it holds

$$\frac{\partial}{\partial v} (-v^{2^r} + 2r(v-1) + 1) = 2r \underbrace{(1 - v^{2^{r-1}})}_{>0} > 0.$$

The second factor of the numerator is monotone increasing in ν and from $\nu \in (0, 1)$ it follows

$$-\nu^{2r} + 2r(\nu - 1) + 1 < -1^{2r} + 2r(1 - 1) + 1 = -1 + 1 = 0.$$

So, the derivative is strictly negative for $\nu \in (0, 1)$.

The smaller the contraction factor γ , the faster is the convergence of the contraction mapping. So, a large ν is desired for a better convergence rate. In terms of an SSO-MDP, this can be interpreted as follows: If there exists an action that can be played almost perfectly, the probability of failure will be minimal. Such an action would lead to a very small ν and a bad contraction factor which would be close to 1.

As we have seen that SSO-MDPs can be converted to discounted MDPs, a summary over the worst-case running times of policy iteration and value iteration regarding the number of arithmetic expressions is presented. It has been shown by Tseng (1990) that for a fixed discount rate value-iteration converges to the optimal policy in polynomial time. Since policy iteration is at least as fast as value-iteration (Puterman, 2005, proof of Thm. 6.4.6), this result can be transferred to policy iteration. This conclusion is made by Littman, Dean, and Kaelbling (1995).

In 2011, Ye proved that

“the classic policy-iteration method [...] and the original simplex method with the most-negative-reduced-cost pivoting rule of Dantzig are strongly polynomial-time algorithms for solving the Markov decision problem (MDP) with a fixed discount rate.” (Ye, 2011)

The same pivoting rule of the simplex method on general linear programs was shown to be exponential by Klee and Minty (1972). So, linear programs resulting from MDPs must obey a specific structure such that this result is possible.

3.5 Linear Programming Formulations

Next, a linear programming formulation suited to the special structure of SSO-MDPs is derived. For this purpose, the input parameters of SSO-MDPs are resumed in matrix-vector notation. Let $n = |S|$ be the finite number of states and $m = |A|$ be the finite number of actions of the considered SSO-MDP. Furthermore, let A' be the augmented action set consisting of state-action pairs

$$\begin{aligned} A' &\subseteq S \times A, \\ A'_s &:= \{a = (s, \tilde{a}) \mid \tilde{a} \in A_s\}. \end{aligned}$$

The augmented action set has at most cardinality $n \cdot m$. Assume in the following, it is exactly $n \cdot m$.

Let $P \in \mathbb{R}^{(n \times m) \times n}$ be the transition matrix, where an entry $P_{a,s}$ equals the transition probability under action $a \in A'$ to state s , which is $p(s|a)$. Since state-action pairs are considered, the “from state” in the transition probability term need not be specified and is included in the action. A second matrix $J \in \{0, 1\}^{(n \times m) \times n}$ is used that states which action is available in which state. So, $J_{a,s} = 1$ if and only if action a is available in state s . For unique action sets, like the considered state-action pairs $a = (s, \tilde{a})$, the matrix J has exactly one single 1 per row which is in the column corresponding to state s . It should be stressed that in this subsection the matrices P and J contain the transitions to the absorbing states and

the artificial actions at the absorbing states. In the setting of Bertsekas-SSP MDPs in Bertsekas, 2001, Ch. 2 this is not the case.

Furthermore, a possibly randomized decision rule d can simply be written as a $n \times m$ -dimensional vector $d \in [0, 1]^{n \times m}$ instead of a function depending on the state s . The component $d(a)$ is the probability that action $a = (s, \tilde{a}) \in \mathcal{A}'$ is chosen under decision rule d :

$$d(a) = d((s, \tilde{a})) := q_s(\tilde{a}).$$

3.5.1 A Primal Linear Programming Formulation

As a starting point for a linear programming formulation, the standard linear programming formulation for positive bounded infinite horizon MDPs, which was presented in Subsection 2.3.7, is used. It would also be possible to use the linear programming formulation of Bertsekas-SSP MDPs since SSO-MDPs belong to this class of MDPs. The linear programming formulation of Bertsekas-SSP MDPs differs from that of POSB MDPs in the way that it has no non-negativity constraints on the variables v and the matrices J and P do not contain the artificial actions. However, in this thesis the formulation for POSB MDPs is used as a starting point to see where the characteristic properties of SSO-MDPs make a difference. Adapted to the matrix-vector notation of this section, the primal linear programming formulation for POSB MDPs equals

$$\left| \begin{array}{l} \min \omega^T v \\ (J - P)v \geq r \\ v \geq 0. \end{array} \right| \quad (\text{primal POSB LP})$$

The primal LP formulation for POSB MDPs has a constraint for each action available in each state and therefore in the current setting of state-action pairs $n \cdot m$ constraints and n variables. The vector $r \in \mathbb{R}^{n \times m}$ is a reward vector containing the expected total reward of performing an action $a \in \mathcal{A}'$. It should be clarified that $(J - P)v \geq r$ is indeed equivalent to

$$v(s) - \sum_{j \in S} p(j|s, \tilde{a})v(j) \geq r(s, \tilde{a}) \quad \forall \tilde{a} \in \mathcal{A}_s, \quad \forall s \in S,$$

which are the inequalities of the primal LP from Section 2.3.7. This can be seen as follows:

$$\begin{aligned} & (J - P)v \geq r \\ \Leftrightarrow & \sum_{j \in S} (J - P)_{a,j} v(j) \geq r(a), \quad \forall a = (s, \tilde{a}) \in \mathcal{A}' \\ \Leftrightarrow & \sum_{j \in S} 1_{a \in \mathcal{A}'_j} v(j) - \sum_{j \in S} p(j|a) v(j) \geq r(a), \quad \forall a = (s, \tilde{a}) \in \mathcal{A}' \\ \Leftrightarrow & v(s) - \sum_{j \in S} p(j|s, \tilde{a}) v(j) \geq r(s, \tilde{a}), \quad \forall \tilde{a} \in \mathcal{A}_s, \quad \forall s \in S. \end{aligned}$$

This formulation is now adapted to the specific properties of an SSO-MDP. As the starting state in SSO-MDPs is known, the objective function $\min \omega^T v$ can be replaced by $\min v_{s_1}$.

Furthermore, all transitions give a reward of 0 except a transition to a winning state, which gives a reward of 1. For a state-action pair $a = (s, \bar{a})$ in a state $s \in S \setminus (W \cup L)$, the inequality

$$v(s) - \sum_{j \in S} p(j|a)v(j) \geq r(a) \quad (3.1)$$

simplifies to

$$\begin{aligned} v(s) - \sum_{j \in S} p(j|a)v(j) &\geq \sum_{j \in W} p(j|a) \\ \Leftrightarrow v(s) - \sum_{j \in S \setminus W} p(j|a)v(j) - \sum_{j \in W} p(j|a)[v(j) + 1] &\geq 0. \end{aligned}$$

Since all losing and all winning states are absorbing states, there exist only artificial actions $\bar{a} \in \mathcal{A}_s$ for $s \in W \cup L$. With probability 1 these actions return to s while generating a reward of zero. So, for all artificial actions in the absorbing state, which means for all $\bar{a} \in \mathcal{A}_s$ with $s \in W \cup L$ inequality 3.1 simplifies to

$$v(s) - v(s) \geq 0.$$

This inequality is always satisfied and constraints belonging to the artificial actions in the absorbing states could be removed from the linear program. However, in this subsection they are left in the program such that the matrices J and P need not be formally redefined.

As next, we make a renaming of the variables: Define

$$\tilde{v}(s) := \begin{cases} v(s) + 1 & \text{for } s \in W \\ v(s) & \text{else.} \end{cases}$$

Then, the linear programming formulation equals

$$\left| \begin{array}{l} \min v(s_1) \\ \tilde{v}(s) - \sum_{j \in S} p(j|s, \bar{a})\tilde{v}(j) \geq 0, \forall \bar{a} \in \mathcal{A}_s \text{ with } s \in S \setminus (W \cup L) \\ \tilde{v}(s) - v(s) \geq 0, \forall \bar{a} \in \mathcal{A}_s \text{ with } s \in W \cup L \\ \tilde{v}(s) = v(s) + 1, \forall s \in W \\ \tilde{v}(s) = v(s), \forall s \in S \setminus W \\ v(s) \geq 0, \forall s \in S. \end{array} \right| \quad (3.2)$$

We notice that in the first inequality set only actions $\bar{a} \in \mathcal{A}_s$ with $s \in S \setminus (W \cup L)$ are considered. So, $\tilde{v}(s)$ for $s \in W \cup L$ may only occur in the sum $-\sum_{j \in S} p(j|s, \bar{a})\tilde{v}(j)$ in the first inequality set. Since $p(j|s, \bar{a})$ and $\tilde{v}(s)$ are non-negative for all $j, s \in S, \bar{a} \in \mathcal{A}$, the \geq -inequality can only become more relaxed if we decrease $\tilde{v}(s)$ for $s \in W \cup L$. So, the objective function will not increase if $v(s)$ is decreased for $s \in W \cup L$. Zero is a lower bound for $v(s), s \in W \cup L$. Therefore, $v(s)$ can be set to 0 for $s \in W \cup L$ without increasing the optimal objective value. The resulting linear programming formulation is

$$\left| \begin{array}{l} \min v(s_1) \\ \tilde{v}(s) - \sum_{j \in S} p(j|s, \bar{a})\tilde{v}(j) \geq 0, \forall \bar{a} \in \mathcal{A}_s \text{ with } s \in S \setminus (W \cup L) \\ \tilde{v}(s) - v(s) \geq 0, \forall \bar{a} \in \mathcal{A}_s \text{ with } s \in W \cup L \\ \tilde{v}(s) = 1, \forall s \in W \\ \tilde{v}(s) = 0, \forall s \in L \\ \tilde{v}(s) \geq 0, \forall s \in S. \end{array} \right| \quad (3.3)$$

Since now $v(s)$ does not occur any more in the linear program, the variables are again denoted by v .

The non-negativity constraints of v in the primal LP of the POSB MDPs guarantee that a minimal solution $v \in V^+ = \mathbb{R}_{\geq 0}$ is found, which corresponds to the value of the POSB MDP. As in SSO-MDPs all policies are proper, for each $\tilde{a} \in \mathcal{A}$, the inequality

$$v(s) - \sum_{j \in S} p(j|s, \tilde{a})v(j) \geq 0, \quad \forall \tilde{a} \in \mathcal{A}, \text{ with } s \in S \setminus (W \cup L)$$

can be transformed by replacing $v(j)$ recursively by this inequality until

$$v(s) \geq \sum_{j \in W \cup L} \sum_{t=1}^{\infty} \mathbb{P}\{X_t = s, Y_t = \tilde{a}, X_{t+1} = j\}v(j), \quad \forall \tilde{a} \in \mathcal{A}, \text{ with } s \in S \setminus (W \cup L).$$

The right side of the inequality is non-negative since $v(s) = 1$ for $s \in W$ and $v(s) = 0$ for $s \in L$. Therefore, the non-negativity conditions of v can be omitted in the primal LP formulation for SSO-MDPs.

Altogether, a valid primal linear programming formulation for SSO-MDPs is:

$$\left| \begin{array}{l} \min v_{s_1} \\ (J - P)v \geq 0 \\ v(s) = 0, \quad \forall s \in L \\ v(s) = 1, \quad \forall s \in W \end{array} \right| \quad (\text{primal LP for SSO-MDPs})$$

This primal linear programming formulation for SSO-MDPs has n variables and $m \cdot n + |W| + |L|$ constraints.

Comparison to other LP formulations

A short comparison of the derived formulation to other primal linear programming formulations of different MDP classes is given.

Let \tilde{P} be the transition matrix restricted to non-goal states and non-artificial actions of non-goal states. Similarly, let \tilde{J} contain only state-action pairs $a = (s, \tilde{a})$ with $\tilde{a} \in \mathcal{A}$, with $s \in S \setminus (W \cup L)$. Then, the standard primal linear program of Bertsekas-SSP MDPs converted to the reward-based setting is

$$\left| \begin{array}{l} \min v_{s_1} \\ (\tilde{J} - \tilde{P})v \geq r. \end{array} \right|$$

Observe that this linear program has only variables $v(s)$ for $s \in S \setminus \{W \cup L\}$ and no non-negativity constraints. In Bertsekas-SSP MDPs, the dynamic programming operator has a unique fixed point and therefore no non-negativity constraints on v are needed. Also the value function of the absorbing states $W \cup L$ is reward free. By some similar transformations, the LP formulation of Bertsekas-SSP MDPs can also be converted into the primal LP formulation SSO-MDPs presented above.

Furthermore, the primal SSO-MDP formulation can be seen as a reward based formulation of the dual linear programm of Guillot and Stauffer (2017) applied to a special MDP class. The dual linear programm of Guillot and Stauffer (2017) is

$$\left| \begin{array}{l} \max \mathbf{1}^T y \\ (J - P)y \leq c, \end{array} \right| \quad \Leftrightarrow \quad \left| \begin{array}{l} -\min \mathbf{1}^T (-y) \\ (J - P)(-y) \geq -c, \end{array} \right|$$

where $c \in \mathbb{R}^n$ is a cost vector and the variables are $y \in \mathbb{R}^n$. The objective value of the SSO-MDP formulation equals the negative of the objective value of Guillot and Stauffer's linear program.

3.5.2 A Dual Linear Programming Formulation

The dual linear program of the primal LP for SSO-MDPs of the last subsection in matrix vector notation is:

$$\left(\begin{array}{l} \max \sum_{s \in \mathcal{W}} w(s) \\ (J - P)^T x = \begin{cases} 1 & s = s_1 \\ -w_s & s \in \mathcal{W} \\ -l_s & s \in L \\ 0 & \text{else.} \end{cases} \\ x \geq 0 \end{array} \right) \quad (\text{dual LP for SSO-MDPs})$$

Dual variables x_a , $a = (s, \bar{a})$, $\forall \bar{a} \in \mathcal{A}_s$ are used for the inequalities $(J - P)v \geq 0$ of the primal LP for SSO-MDPs. For the last two sets of constraints, which define the value function of the winning and losing states, the dual variables w_s , $\forall s \in \mathcal{W}$ and l_s , $\forall s \in L$ are introduced. This linear programming formulation has n rows and $m \cdot n + |\mathcal{W}| + |L|$ variables.

The dual LP formulation contains constraints that can be interpreted as flow constraints. This interpretation is also used by Littman, Dean, and Kaelbling (1995) who write

“Under this interpretation, the constraints are flow conservation constraints that say that the total flow exiting state j is equal to the flow beginning at state j (always 1) plus the flow entering state j via all possible combinations of states and actions weighted by their probability” (Littman, Dean, and Kaelbling, 1995).

The difference to an ordinary flow condition is the inclusion of the transition probabilities. The transition probabilities distribute the incoming flow of an action to the states. This flow distribution by the transition probabilities can be viewed as exogenous flow variables $y_{a,s} := p(s|a)x_a$, $\forall a \in \mathcal{A}'$, $s \in \mathcal{S}$ that specify the outgoing flow of an action and can not be influenced by the decision maker. In an ordinary flow problem, the flow conservation constraints would be

$$\sum_{a \in \mathcal{A}'_s} x_a - \sum_{a \in \mathcal{A}'} y(s|a) = 0 \quad \forall s \in \mathcal{S} \setminus (\{s_1\} \cup \mathcal{W} \cup L) \quad (3.4)$$

$$\sum_{s \in \mathcal{S}} y(s|a) - x_a = 0 \quad \forall a \in \mathcal{A}'. \quad (3.5)$$

Since in an MDP the transition probabilities define a probability distribution the second equation is always satisfied:

$$\sum_{s \in \mathcal{S}} y_{a,s} - x_a = \sum_{s \in \mathcal{S}} p(s|a)x_a - x_a = x_a - x_a = 0.$$

So, we do not need to incorporate flow constraints for actions in the linear program as the transition probabilities determine which fraction of the incoming flow x_a goes to which state.

A stationary policy d^∞ can be determined from the flow variables in the following way: For a state s with $\sum_{\bar{a} \in A_s} x_{s,\bar{a}} > 0$ define

$$d(a) = d((s, \bar{a})) := \frac{x_{s,\bar{a}}}{\sum_{i \in A_s} x_{s,i}}.$$

In all other states, define d as an arbitrary probability distribution over the set of available actions. This is according to Definition 2.3.29 of the previous chapter. So, $d(a)$ is the fraction of the flow in a state node that goes out to an action node. If the complete flow in a state node goes out to one single action node, we will get a deterministic policy d .

Littman, Dean, and Kaelbling write in Littman, Dean, and Kaelbling (1995) that the objective value of the dual linear program can be interpreted as the expected total costs – which equals in the setting of this thesis the expected total reward – of this stationary policy. Furthermore, they state how a deterministic optimal policy can be computed from a dual solution. The following equation is an adaptation of Littman, Dean, and Kaelbling notation to the notation of this thesis. Furthermore, the computation of a deterministic optimal policy was expanded to the case where $|\arg \max_{d' \in A_s} x_{s,d'}| > 1$:

$$d(a) = d((s, \bar{a})) = \begin{cases} 1 & \text{if } \bar{a} \in \arg \max_{d' \in A_s} x_{s,d'} \text{ and } d((s, k)) = 0 \forall k \in \arg \max_{d' \in A_s} x_{s,d'} \setminus \{\bar{a}\}, \\ 0 & \text{else.} \end{cases}$$

In the next Subsection 3.6, the connection between a realization of the Markov process under a certain policy and the flow in the dual LP is examined in detail. But first, an example of an SSO-MDP, its dual LP formulation and the derived stationary policy from an optimal solution of the dual LP is presented.

Example 3.1:

Figure 3.3 shows a slightly modified version of the example in Figure 3.1 from the beginning of this section. The difference from the original example problem is that it contains a cycle, namely (s_2, a_3, s_2) .

In the Example presented by Figure 3.3, the number of actions m equals 6 and then number of states n equals 4. For the ordering $(a_1, a_2, a_3, a_4, a_w, a_l)$ of the actions and (s_1, s_2, w, l) of the states, the matrices of the dual LP for SSO-MDPs formulation can be specified as

$$P = \begin{pmatrix} 0 & 0.6 & 0.4 & 0 \\ 0 & 0.6 & 0 & 0.4 \\ 0 & 0.5 & 0.2 & 0.3 \\ 0 & 0 & 0.6 & 0.4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad J = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Note that the artificial actions in the absorbing states w and l could be removed from the matrices and need not be included in the model as they are redundant. However, to be consistent with the definitions of P and J of this subsection, they are included in this formulation.

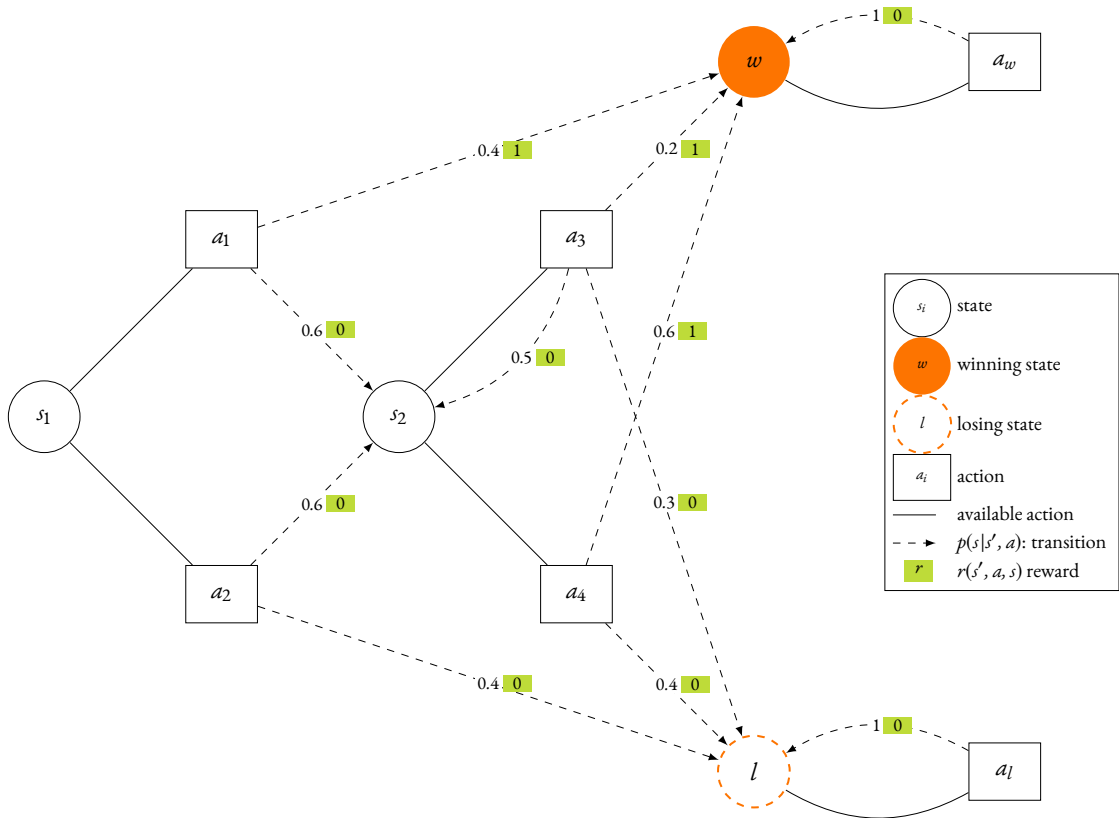


Figure 3.3: SSO-MDP Example

The dual LP for SSO-MDPs formulation applied to this example is

$$\left| \begin{array}{c} \max w \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ -0.6 & -0.6 & 0.5 & 1 & 0 & 0 \\ -0.4 & 0 & -0.2 & -0.6 & 0 & 0 \\ 0 & -0.4 & -0.3 & -0.4 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_{a_1} \\ x_{a_2} \\ x_{a_3} \\ x_{a_4} \\ x_{a_w} \\ x_{a_l} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -w \\ -l \end{pmatrix} \\ x \geq 0 \end{array} \right|$$

An optimal solution of this linear program is

$$\begin{pmatrix} x_{a_1} \\ x_{a_2} \\ x_{a_3} \\ x_{a_4} \\ x_{a_w} \\ x_{a_l} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0.6 \\ 0 \\ 0 \end{pmatrix}, w = 0.76, l = 0.24$$

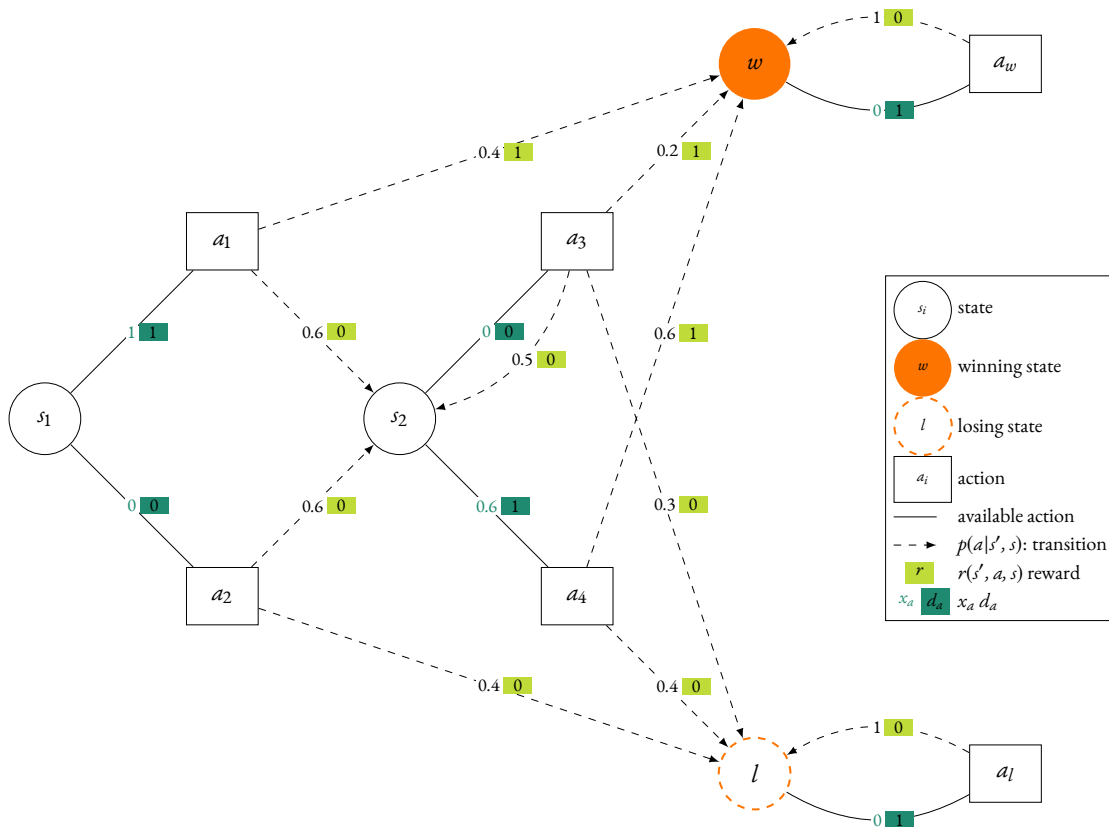


Figure 3.4: Solution of SSO-MDP Example

with an objective value of 0.76. This translates into the optimal decision rule

$$d = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}.$$

The solution is presented in Figure 3.4. The first blue number denotes the action flow x_a and the second number the derived decision rule $d(a)$. The optimal decision rule selects action a_1 in s_1 and action a_4 in s_1 . *

3.6 Flow Networks associated with SSO-MDPs

In the last section, the dual variables of the dual LP for SSO-MDPs were interpreted as flow variables. The goal of this section is to define a flow network for SSO-MDPs and to clarify how a flow described by the dual linear programming formulation corresponds to a realization of a Markov process under a

given decision rule. This section is relatively long and technical. The main result is the static maximum flow formulation 3.12 which is equivalent to the dual linear programming formulation seen in the last section. However, starting from scratch in an own setup, Theorem 3.6.23 proves that the linear program can be used for finding an optimal decision rule of the SSO-MDP without relying on any other result of linear programming formulations for MDPs.

In general a flow in a network, as for example a solution of the maximum flow problem (Definition 2.4.8), assigns a non-negative flow value $x_{i,j}$ to each edge such that the flow conditions and capacity restrictions are met. The flow values $x_{i,j}$ are not time depended, instead they describe a static flow that can be send from the sink node to the source node. At each point in time, the flow per time unit is $x_{i,j}$ on edge (i, j) .

In contrast, an SSO-MDP relies on a stochastic process that evolves over time. The random process consists of a random variable X_t that describes the state of the system at time t and a second random variable Y_t that describes which action is selected at time t according to a, possibly randomized, decision rule $Y_t = d(X_t)$. If the decision rule is deterministic, $d(X_t)$ is also deterministic and does not have to be expressed by a separate random variable.

3.6.1 Basic Definitions

Graphical representations of SSO-MDPs were already used in the examples like, e.g., in Figure 3.1. As indicated, unique actions are necessary for a clearly arranged graphical representation. If actions are available in different states and the transition probabilities differ depending on the state in which the action is chosen, it would not be possible to associate a single transition probability with the edges that connect the action with its successors. Therefore, in this section an augmented action set \mathcal{A} consisting of state-action pairs is assumed.

Before defining a flow network associated to an SSO-MDP under a policy $\pi \in \Pi^{MR}$, a graph associated to an SSO-MDP is formally defined as:

Definition 3.6.1 (Graph associated with an SSO-MDP).

An associated graph $G = (N, E, \beta)$ of an SSO-MDP

$$(S, \mathcal{A}, p(\cdot|s, a), r(s, a), W, L, s_1)$$

is a directed graph with a node set $N := \{1, \dots, |S \cup \mathcal{A}|\}$ and a bijection β that maps the states and actions of the SSO-MDP to nodes of the graph:

$$\begin{aligned} \beta : S \cup \mathcal{A} &\rightarrow N \\ i &\mapsto \beta(i). \end{aligned}$$

The edge set $E \subseteq N \times N$ is defined as

$$\begin{aligned} E := &\{(i, j) : \beta^{-1}(i) = s \in S, \beta^{-1}(j) \in \mathcal{A}_s\} \\ &\cup \{(i, j) : \beta^{-1}(i) = a \in \mathcal{A}, \beta^{-1}(j) = s \in S, p(s|\mathcal{A}^{-1}(a), a) > 0\}. \end{aligned}$$

By definition, in an associated graph of an SSO-MDP, there exist only edges between nodes that correspond to states and nodes that correspond to actions in the MDP. Therefore, an associated graph of an SSO-MDP is always a bipartite graph, where N splits into two disjoint sets of nodes $N = N_A \cup N_S$ with

$$\begin{aligned} N_A &:= \{n \in N \mid \beta^{-1}(n) \in A\} \\ N_S &:= \{n \in N \mid \beta^{-1}(n) \in S\}. \end{aligned}$$

Observe that no two nodes in N_A , respectively N_S , are adjacent.

Observation 3.6.2:

Let G be an associated graph of an SSO-MDP, then G is a bipartite graph.

The nodes $n \in N_A$ are called *action nodes* and the nodes $n \in N_S$ are called *state nodes*. Note that most of the time, the artificial action nodes in the absorbing states will not be explicitly named or drawn. However, they are included in E . The set of winning states is denoted by the symbol N_S^W and the set of losing states by N_S^L .

Following definition 2.4.7, a flow network is a directed graph where nodes or arcs have associated numerical values. For the flow network of an associated graph of an SSO-MDP, for all edges $e \in E$ an edge weight $\alpha(e) \in [0, 1]$ is defined. The terms are called “edge weight” because they specify a share of flow: For an edge $e = (i, j)$ the edge weight $\alpha(e)$ is the fraction of flow in i that leaves i over edge e . The edge weight of an edge from a state node to an action node is defined to be 1 while the edge weight of an edge from an action node to a state node is derived from the transition probabilities of the SSO-MDP. The following definition specifies the edge weights α formally.

Definition 3.6.3 (A Flow Network associated with an SSO-MDP).

The flow network of an associated graph $G = (N, E, \beta)$ of an SSO-MDP is a tuple (N, E, β, α) where the edge weights α are defined as

$$\alpha(i, j) := \begin{cases} 1, & \text{if } (i, j) \in E, \beta^{-1}(i) \in S, \beta^{-1}(j) \in A, \\ p(s|A^{-1}(a), a), & \text{if } (i, j) \in E, \beta^{-1}(i) = a \in A, \beta^{-1}(j) = s \in S. \end{cases}$$

Later, the edge weight of a path in the graph is needed. The edge weight of a path from u to v is the fraction of flow available in u that follows the path and enters v . Therefore, the edge weight of a path $\xi_{u,v}$ must be the product of all edge weights occurring on that path:

$$\alpha(\xi_{u,v}) := \prod_{e \in \xi_{u,v}} \alpha(e).$$

In the flow network of an associated graph of an SSO-MDP, there are no capacity restrictions on the edges nor costs for traversing an edge. The reward structure of an SSO-MDP is uniquely defined by the set of winning states. So, there is no need to encode a reward function in the flow network. For a given starting state s_1 , a set of winning states W and a set losing states L , the flow network (N, E, β, α) uniquely defines an SSO-MDP.

In the following example, a simple SSO-MDP is considered and the flow network of an associated graph of that SSO-MDP is presented.

Example 3.2 (Flow Network of an associated graph of an SSO-MDP):

Consider an SSO-MDP with three states $S = \{s_1, w, l\}$. Assume the states w and l are absorbing states such that w is a winning state and l is a losing state. In this example and in the following of this thesis, the explicit notation and drawing of the artificial action nodes in the absorbing states is left out. Instead, the winning states and losing states are colored in red to distinguish them from ordinary transient states. Let s_1 be a state where an action a_1 is available. Furthermore, let the transitions probabilities after choosing action a_1 in s_1 be:

$$p(w|s_1, a_1) = 0.2 \quad p(l|s_1, a_1) = 0.2 \quad p(s_1|s_1, a_1) = 0.6.$$

Then, an associated graph of this example MDP is $G = (N, E, \beta)$ with $N = \{1, 2, 3, 4\}$ and

$$\begin{array}{ll} \beta(s_1) = 1 & \beta(a_1) = 2 \\ \beta(l) = 3 & \beta(w) = 4. \end{array}$$

The edge set equals $E = \{(1, 2), (2, 1), (2, 3), (2, 4)\}$. According to the definition of an associated flow network, we get for the edge weights

$$\begin{array}{ll} \alpha(1, 2) = 1 & \alpha(2, 1) = 0.6 \\ \alpha(2, 3) = 0.2 & \alpha(2, 4) = 0.2. \end{array}$$

Figure 3.5 is a graphical illustration of the flow network (N, E, β, α) defined in this example. *

3.6.2 Induced Flow of an SSO-MDP

As next, it is shown how the random process induced by an SSO-MDP and a stationary policy $\pi = d^\infty$ corresponds to a flow in an associated flow network of that SSO-MDP.¹

To define a random process induced by a policy d^∞ in the flow network of an associated graph of an SSO-MDP, a random variable Z_t that takes values in the node set N is introduced:

¹ The induced stochastic process of an MDP under policy π is explained in Section 2.1.

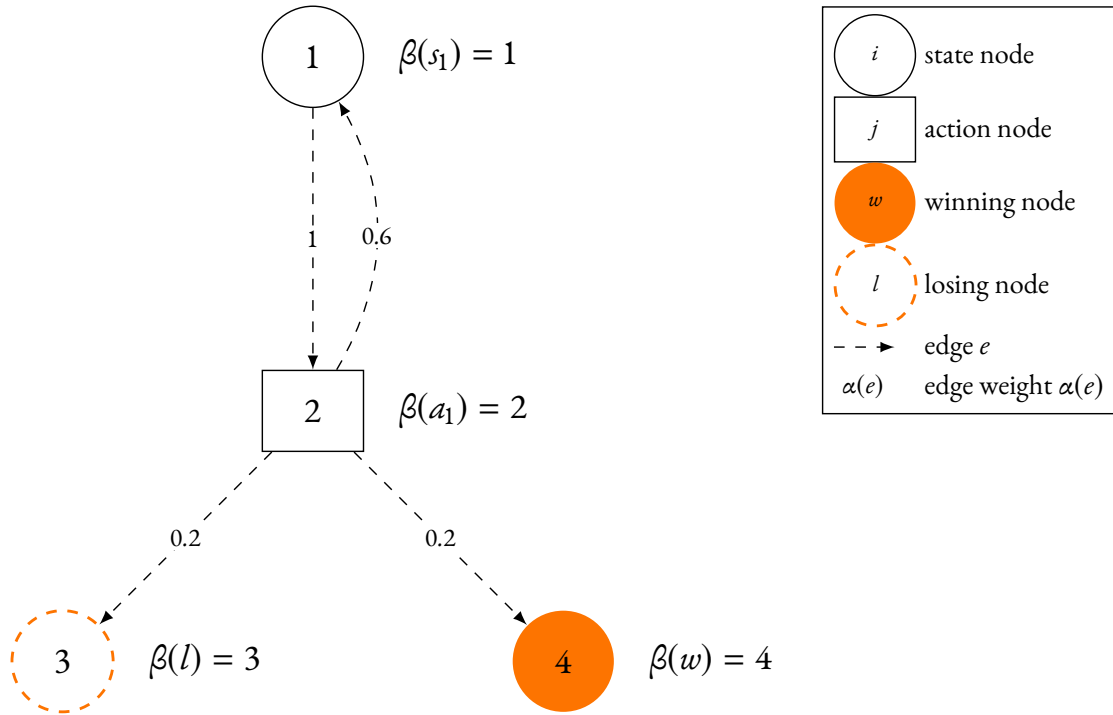


Figure 3.5: Flow network example of an SSO-MDP

Definition 3.6.4 (Random Variable of the flow network).

Let (Ω, F, \mathbb{P}) be a probability space of an MDP as described in Subsection 2.1.1. Define Z_t as a random variable that maps $\omega = (s_1, a_1, s_2, \dots) \in \Omega$ to the measure space $(\mathcal{N}, 2^{\mathcal{N}})$ by

$$Z_t : \Omega \rightarrow \mathcal{N}$$

$$Z_t(\omega) = \begin{cases} \beta(s_{(t+1)/2}) & \text{if } t \equiv 1 \pmod{2}, \\ \beta(a_{t/2}) & \text{if } t \equiv 0 \pmod{2}. \end{cases}$$

So, $Z_1(\omega) = \beta(s_1)$, $Z_2(\omega) = \beta(a_1)$, $Z_3(\omega) = \beta(s_2)$ and so on. Obviously, the random variable Z_t is alternating between nodes in \mathcal{N}_S and \mathcal{N}_A .

A realization of the random variable Z_t can be interpreted as a random walk in the associated flow network of an SSO-MDP. To visualize this random walk over time in a network, a time-expanded flow network of an SSO-MDP is used. In a time-expanded network, the node set is copied for each point in time t and for each edge $(i, j) \in E$ there exists an edge from the node i at time t to the node j at time $t + 1$.

Definition 3.6.5 (Time-Expanded Flow Network of an SSO-MDP).

Assume an associated flow network $(\mathcal{N}, E, \beta, \alpha)$ of an SSO-MDP. Then, the time-expanded flow network $(\mathcal{N}^t, E^t, \beta^t, \alpha^t)$ is defined as

$$\mathcal{N}^t := \{(t, i) \mid t \in \mathbb{N}, i \in \mathcal{N}\}$$

$$\begin{aligned}
E^t &:= \{(t, i), (t+1, j) \mid t \in \mathbb{N}, (i, j) \in E\} \\
\beta^t &:= \begin{cases} \mathbb{N} \times \{S \cup \mathcal{A}\} & \rightarrow N^t \\ (t, i) & \mapsto (t, \beta(i)); \end{cases} \\
\alpha^t &:= \begin{cases} E^t & \rightarrow [0, 1] \\ \alpha^t((t, i), (t+1, j)) & \mapsto \alpha(i, j); \end{cases}
\end{aligned}$$

Example 3.3 (Random walk in flow network):

Consider the SSO-MDP of Example 3.2. Assume the SSO-MDP starts in s_1 and uses a decision rule d that selects action a_1 in state s_1 . A possible realization of the Markov chain resulting from this MDP and the decision rule d is

$$\omega = (s_1, a_1, s_1, a_1, w, \dots).$$

This realization ends in the winning state after two transitions and stays there forever. Accordingly, the random variable Z_t takes the values

$$Z_1 = 1 \quad Z_2 = 2 \quad Z_3 = 1 \quad Z_4 = 2 \quad Z_5 = 4 \quad \dots$$

The time-expanded network of this example is presented in Figure 3.6. The realization of this example corresponds to a walk along the following nodes $(t, i) \in N^t$:

$$(1, 1) - (2, 2) - (3, 1) - (4, 2) - (5, 4) - (6, 4) \dots \quad *$$

Z_1 is by definition of Z_t always a state node. Since Z_t is alternating between state and action nodes and those sets are disjoint, it can be concluded from the value of Z_t whether t is even or not. And the other way around, it can be deduced from t whether $Z_t \in N_S$ or $Z_t \in N_A$ holds by using the definition of Z_t . So,

$$\begin{aligned}
Z_t \in N_S &\Leftrightarrow t \equiv 1 \pmod{2} \\
Z_t \in N_A &\Leftrightarrow t \equiv 0 \pmod{2}.
\end{aligned}$$

In an MDP, we are interested in the expected total reward of a decision rule instead of the reward of one realization. Proposition 3.6.6 relates the value of a policy d^∞ in an SSO-MDP to the probability distribution of the random variable Z_t in the flow network:

Proposition 3.6.6 (Expected total reward of SSO-MDP and Z_t):

The value of a policy d^∞ in an SSO-MDP starting in a state $s \in S \setminus \{W \cup L\}$ equals the probability that Z_t enters a node in N_S^W :

$$v^{d^\infty}(s) = \sum_{t=2}^{\infty} \mathbb{P}^{d^\infty} \{Z_{2t-3} \notin N_S^W, Z_{2t-1} \in N_S^W \mid Z_1 = \beta(s)\}.$$

PROOF. The expected total reward of a stationary policy d^∞ equals:

$$\begin{aligned}
v^{d^\infty}(s) &= \mathbb{E}_s^{d^\infty} \left\{ \sum_{t=1}^{\infty} r(X_t, Y_t) \right\} \\
&\stackrel{\text{Prop. 3.4.1}}{=} \sum_{t=2}^{\infty} \mathbb{P}^{d^\infty} \{X_{t-1} \notin W, X_t \in W \mid X_1 = s\} \\
&= \sum_{t=2}^{\infty} \mathbb{P}^{d^\infty} \{(s_1, a_1, s_2, \dots) \in \Omega : s_{t-1} \notin W, s_t \in W \mid s_1 = s\} \\
&\stackrel{\text{Def. } Z_t}{=} \sum_{t=2}^{\infty} \mathbb{P}^{d^\infty} \{\beta^{-1}(Z_{2t-3}) \notin W, \beta^{-1}(Z_{2t-1}) \in W \mid \beta^{-1}(Z_1) = s\} \\
&= \sum_{t=2}^{\infty} \mathbb{P}^{d^\infty} \{Z_{2t-3} \notin N_S^W, Z_{2t-1} \in N_S^W \mid Z_1 = \beta(s)\}.
\end{aligned}$$

The result directly follows from Proposition 3.4.1 and the definition of Z_t . ■

The last proposition related the probability distribution of the random variable Z_t to the expected total reward of a decision rule. This motivates to examine the distribution of Z_t under a policy d^∞ .

As already observed, Z_t is alternating between state nodes and action nodes, which are disjoint sets. We know that for t even, Z_t takes a value in N_A independent of the used policy d^∞ . Hence, the probability $\mathbb{P}\{Z_t \in N_S\}$ is zero for every even time point $t = 2k$, $k \in \mathbb{N}$. Also, the probability $\mathbb{P}\{Z_t \in N_A\}$ is zero for every odd time point $t = 2k + 1$, $k \in \mathbb{N}_0$ independent of the used policy d^∞ .

Observation 3.6.7:

Let Z_t be a random variable in the flow network of an associated graph of an SSO-MDP that is constructed after Definition 3.6.4. Then,

$$\begin{aligned}
\mathbb{P}^{d^\infty} \{Z_{2k} \in N_S\} &= 0, \quad \forall k = 1, 2, 3, \dots, \quad \forall d^\infty \in \Pi^{MR} \\
\mathbb{P}^{d^\infty} \{Z_{2k+1} \in N_A\} &= 0, \quad \forall k = 0, 1, 2, \dots, \quad \forall d^\infty \in \Pi^{MR}.
\end{aligned}$$

The goal is to calculate the probability that the random walk Z_t is at node j at time t when the process has started in node i at time $t' \leq t$ and policy d^∞ is used. So, we want to determine

$$\mathbb{P}^{d^\infty} \{Z_t = j \mid Z_{t'} = i\} \text{ for } t', t \in \mathbb{N} \text{ with } t \geq t'.$$

The edge weight α of state-action edges can be adapted to the used policy d^∞ such that the mentioned probability equals the sum of the weights of all paths from (t', i) to (t, j) in the time-expanded network associated to the SSO-MDP.

Proposition 3.6.8 (Probability distribution of Z_t under d^∞):

Let $d^\infty \in \Pi^{MR}$ be a stationary Markovian policy and P_1 a distribution of the initial state. Define the adapted edge weight function α_{d^∞} under d^∞ as

$$\alpha_{d^\infty}(i, j) := \begin{cases} q_{d(s)}(a) \cdot \alpha(i, j), & \text{if } (i, j) \in E, \beta^{-1}(i) = s \in S, \beta^{-1}(j) = a \in A, \\ \alpha(i, j), & \text{else.} \end{cases}$$

Let $\alpha_{d^\infty}^t$ be the time-expanded adapted edge weights in the time-expanded graph based on α_{d^∞} .

Then, the probability that the random variable Z_t is at node j at time t when the process has started in node i at time t' and a stationary policy d^∞ is used equals the sum of the weights of all paths from (t', i) to (t, j) in the adapted time-expanded network $(N^t, E^t, \beta^t, \alpha_{d^\infty}^t)$ of the SSO-MDP:

$$\begin{aligned} & \mathbb{P}^{d^\infty} \{Z_t = j \mid Z_{t'} = i\} \\ &= \sum_{\substack{\xi \text{ path from } (t', i) \text{ to } (t, j) \\ \text{in } (N^t, E^t)}} \left[\prod_{e \in \xi} \alpha_{d^\infty}^t(e) \right], \quad \forall d^\infty \in \Pi^{MR}, \end{aligned}$$

for all $t, t' \in \mathbb{N}$ with $t \geq t'$ and $\mathbb{P}^{d^\infty} \{Z_{t'} = i\} > 0$.

PROOF. Since Z_t is alternating between state and action nodes, a case distinction whether t and t' are even or not can be made. Assume $t \equiv 1 \pmod{2}$ and $t' \equiv 1 \pmod{2}$. Then, it holds $\beta^{-1}(Z_t) \in N_S$ as well as $\beta^{-1}(Z_{t'}) \in N_S$ and

$$\begin{aligned} & \mathbb{P}^{d^\infty} \{Z_t = j \mid Z_{t'} = i\} \\ &= \frac{\mathbb{P}^{d^\infty} \{Z_{t'} = i, Z_t = j\}}{\mathbb{P}^{d^\infty} \{Z_{t'} = i\}} \\ &= \frac{\mathbb{P}^{d^\infty} \{(s_1, a_1, s_2, \dots) \in \Omega : s_{(t'+1)/2} = \beta^{-1}(i), s_{(t+1)/2} = \beta^{-1}(j)\}}{\mathbb{P}^{d^\infty} \{(s_1, a_1, s_2, \dots) \in \Omega : s_{(t'+1)/2} = \beta^{-1}(i)\}} \quad (3.6) \\ &= \frac{\sum_{\{\omega \in \Omega : s_{(t'+1)/2} = \beta^{-1}(i), s_{(t+1)/2} = \beta^{-1}(j)\}} P_1(s_1) \cdot q_{d(s_1)}(a_1) \cdot p(s_2|s_1, a_1) \cdot \dots}{\sum_{\{\omega \in \Omega : s_{(t'+1)/2} = \beta^{-1}(i)\}} P_1(s_1) \cdot q_{d(s_1)}(a_1) \cdot p(s_2|s_1, a_1) \cdot \dots} \end{aligned}$$

The definition of Z_t is inserted in Equation 3.6.

In the last equation, the probability of an elementary element $\omega \in \Omega$ is inserted, which is explained in Section 2.1. The sum of the numerator contains all events where the system is in $\beta(i)$ at time $(t'+1)/2$ and in $\beta(j)$ at time $(t+1)/2$. For a given history until $(t+1)/2$ that fulfills this condition, all possible extensions are included in the sum. Since the conditional probabilities of all extensions sum up to 1, the sum can be restricted to all histories up to $(t+1)/2$ that fulfill our requirements. An analogous simplification is possible for the denominator and the fraction simplifies to:

$$= \left[\begin{array}{c} \sum_{\substack{\{(s_1, a_1, \dots, s_{(t+1)/2}) \in (S \times \mathcal{A})^{(t-1)/2} \times S : \\ s_{(t'+1)/2} = \beta^{-1}(i), s_{(t+1)/2} = \beta^{-1}(j)\}} P_1(s_1) \cdot q_{d(s_1)}(a_1) \cdot p(s_2|s_1, a_1) \cdot q_{d(s_2)}(a_2) \cdot \dots \end{array} \right] \cdot \left[\begin{array}{c} \sum_{\substack{\{(s_1, a_1, \dots, s_{(t'+1)/2}) \in (S \times \mathcal{A})^{(t'-1)/2} \times S : \\ s_{(t'+1)/2} = \beta^{-1}(i)\}} P_1(s_1) \cdot q_{d(s_1)}(a_1) \cdot p(s_2|s_1, a_1) \cdot q_{d(s_2)}(a_2) \cdot \dots \end{array} \right]^{-1}$$

Every history that is in $\beta^{-1}(i)$ at time $(t' + 1)/2$ is included in the denominator. In the numerator, each of these histories is extended by all sample paths that go from $\beta^{-1}(i)$ to $\beta^{-1}(j)$ in $(t - t')/2$ time steps. So, the sum of all histories that are at time $(t' + 1)/2$ in $\beta^{-1}(i)$ can be excluded in the sum of the numerator. After the division by the denominator, it remains:

$$\sum_{\substack{\{(s_{(t'+1)/2}, a_{(t'+1)/2}, \dots, s_{(t+1)/2}) \in (S \times \mathcal{A})^{(t-t')/2} \times S : \\ s_{(t'+1)/2} = \beta^{-1}(i), s_{(t+1)/2} = \beta^{-1}(j)\}} qd_{(s_{(t'+1)/2})}(a_{(t'+1)/2}) \cdot \dots \cdot p_{(s_{(t+1)/2} | s_{(t-1)/2}, a_{(t-1)/2})}.$$

The probability of a sample path of the last sum equals the adapted edge weight $\alpha_{d^\infty}^t$ of a path in the time-expanded network that goes from i to j in $t' - t$ time steps. This is true since the time extend network is circle free and the edge weight of a path is defined as the product of the edge weights of edges on that path. Therefore, the last sum equals

$$\begin{aligned} & \sum_{\substack{\mathcal{g} \text{ path from } (t', i) \text{ to } (t, j) \\ \text{in } (N^t, E^t)}} \left[\prod_{e \in \mathcal{g}} \alpha_{d^\infty}^t(e) \right] \\ &= \sum_{\substack{\mathcal{g} \text{ path from } (t', i) \text{ to } (t, j) \\ \text{in } (N^t, E^t)}} \alpha_{d^\infty}^t(\mathcal{g}) \end{aligned}$$

In case that t or t' is even, the same argumentation holds. The only difference is that for t even, Z_t must be in $N_{\mathcal{A}}$ and our sample paths end with an action $a_{t/2}$. ■

Example 3.4 (Probability distribution of Z_t):

Consider the SSO-MDP of Example 3.2 and assume the decision rule $d(s_1) = a_1$. Since a_1 is the only action available in s_1 and d chooses a_1 with certainty, the adapted weight function $\alpha_{d^\infty}^t$ is identical to α^t .

Let Z_t be the random variable according to Definition 3.6.4. Let s_1 be the starting state of the SSO-MDP. Since $\beta(s_1) = 1$, the probability that the random walk Z_t is at node i when the process has started in node 1 at time $t = 1$ is considered.

In Figure 3.6, the probability distribution of Z_t in the adapted time-expanded network for $t = 1, 2, 3, 4, 5$ is denoted. If the probability $\mathbb{P}^{d^\infty} \{Z_t = i \mid Z_1 = 1\}$ is greater zero, the node is highlighted in green and the probability is denoted as a small number at the bottom of the node. *

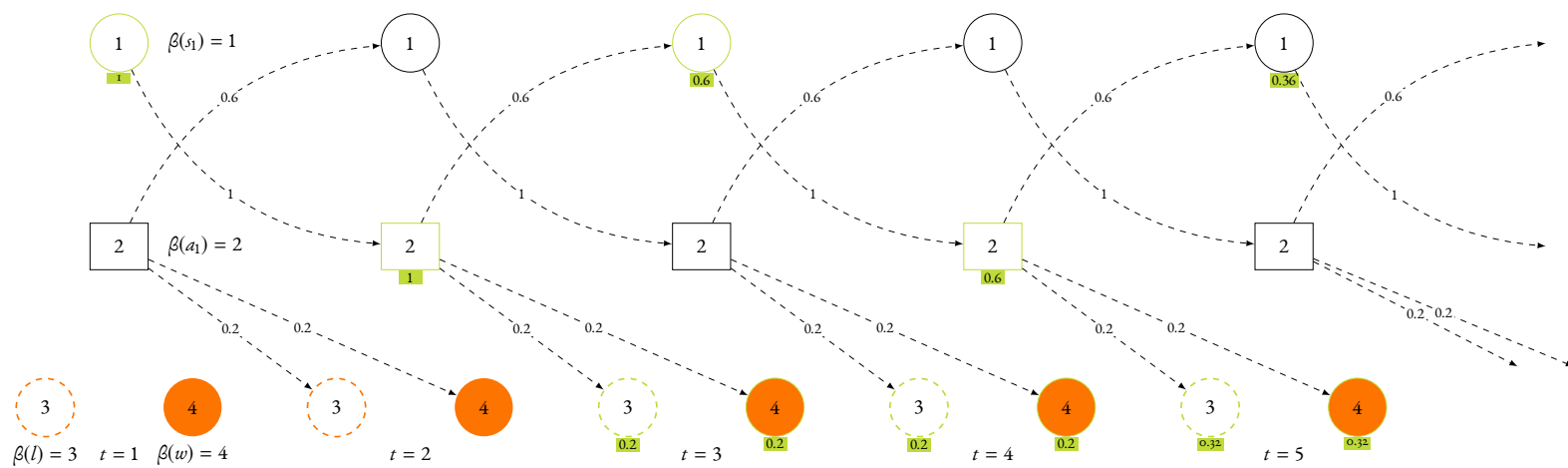


Figure 3.6: Flow generated by random process of SSO-MDP in time-extended network

In the time-expanded network of the flow network associated with an SSO-MDP, the probability distribution of Z_t generates a flow in that network. First, time-dependent flow variables $x_{i,j}^t$ are defined that are induced by Z_t in the time-expanded flow network and show afterwards that they are a valid flow in the time-expanded network.

Definition 3.6.9 (Time varying flow induced by random variable Z_t).

Let $t \in \mathbb{N}$ be a point in time and (i, j) be an edge in E . Then,

$$x_{i,j}^t := \mathbb{P}^{d^\infty} \{Z_{t+1} = j, Z_t = i\}$$

is the time-dependent flow induced by the distribution of the random variable Z_t .

The goal is to show that $\{x_{i,j}^t, t \in \mathbb{N}, (i, j) \in E\}$ is a valid flow in the time-expanded network of an SSO-MDP. By a valid flow, it is meant that $\{x_{i,j}^t, t \in \mathbb{N}, (i, j) \in E\}$ satisfies the flow conservation constraint in each node. The flow condition requires that in each node the amount of incoming flow equals the amount of outgoing flow, see Section 2.4 for more on general flow networks. Intuitively, it might be clear that $x_{i,j}^t$ is a valid flow: By construction of the flow network and the definition of Z_t , a realization of $\{Z_t, t \in \mathbb{N}\}$ is a random walk in the time-expanded network and Z_t moves only along the edges in the time-expanded network. By Proposition 3.6.8 the probability distribution of Z_t is related to path weights in the network. No probability mass of a random variable can be lost or generated. It shifts from one node to the next according to the distribution of Z_t under d^∞ respectively the edge weight $\alpha_{d^\infty}^t$.

However, this should be proven formally on the next few pages even if only technical reformulations are mainly required. Proposition 3.6.10 shows that the sum of outgoing flow in a node at time t equals the probability that Z_t is at that node. Also the sum of incoming flow in a node at time $t - 1$ equals the probability that Z_t is at that node.

Proposition 3.6.10 (Probability of Z_t being at a node):

Let $x_{i,j}^t$ be defined according to definition 3.6.9 which is the induced time-dependent flow by the random variable Z_t . Then,

$$\begin{aligned} \sum_{j:(i,j) \in E} x_{i,j}^t &= \mathbb{P}^{d^\infty} \{Z_t = i\}, \quad \forall t \in \mathbb{N}, \quad \forall i \in N, \\ \sum_{j:(j,i) \in E} x_{j,i}^{t-1} &= \mathbb{P}^{d^\infty} \{Z_t = i\}, \quad \forall t \in \mathbb{N}_{\geq 2}, \quad \forall i \in N. \end{aligned}$$

holds.

PROOF. Let $t \in \mathbb{N}$. Then,

$$\sum_{j:(i,j) \in E} x_{i,j}^t \stackrel{\text{Def.3.6.9}}{=} \sum_{j:(i,j) \in E} \mathbb{P}^{d^\infty} \{Z_{t+1} = j, Z_t = i\}.$$

For $\mathbb{P}^{d^\infty}\{Z_t = i\} = 0$, we have that $\mathbb{P}^{d^\infty}\{Z_{t+1} = j, Z_t = i\} = 0$ for all $(i, j) \in E$ and the statement holds. Otherwise, it holds

$$\sum_{j:(i,j) \in E} \mathbb{P}^{d^\infty}\{Z_{t+1} = j, Z_t = i\} = \sum_{j:(i,j) \in E} \mathbb{P}^{d^\infty}\{Z_{t+1} = j \mid Z_t = i\} \cdot \mathbb{P}^{d^\infty}\{Z_t = i\}.$$

If $i \in N_S$ holds, it is known that

$$\mathbb{P}^{d^\infty}\{Z_{t+1} = j \mid Z_t = i\} = q_{d(s)}(a), \text{ for } \beta(s) = i, \beta(a) = j.$$

Since d specifies a probability distribution over all actions available in s and for each of those actions $a = \beta^{-1}(j)$ an edge $(i, j) \in E$ exists, it holds

$$\begin{aligned} & \sum_{j:(i,j) \in E} \mathbb{P}^{d^\infty}\{Z_{t+1} = j \mid Z_t = i\} \cdot \mathbb{P}^{d^\infty}\{Z_t = i\} \\ &= \sum_{j:(i,j) \in E} q_{d(s)}(a) \cdot \mathbb{P}^{d^\infty}\{Z_t = i\}, \text{ for } \beta(s) = i, \beta(a) = j \\ &= \mathbb{P}^{d^\infty}\{Z_t = i\} \cdot \sum_{j:(i,j) \in E} q_{d(s)}(a), \text{ for } \beta(s) = i, \beta(a) = j \\ &= \mathbb{P}^{d^\infty}\{Z_t = i\}. \end{aligned}$$

If $i \in N_A$ holds, it is known that

$$\mathbb{P}^{d^\infty}\{Z_{t+1} = j \mid Z_t = i\} = p(s \mid \mathcal{A}^{-1}(a), a), \text{ for } \beta(a) = i, \beta(s) = j.$$

Since $p(\cdot \mid \mathcal{A}^{-1}(a), a)$ specifies a probability distribution over all transitions possible from a and for each of those transitions to a state $s = \beta^{-1}(j)$ an edge $(i, j) \in E$ exists, we get

$$\begin{aligned} & \sum_{j:(i,j) \in E} \mathbb{P}^{d^\infty}\{Z_{t+1} = j \mid Z_t = i\} \cdot \mathbb{P}^{d^\infty}\{Z_t = i\} \\ &= \sum_{j:(i,j) \in E} p(s \mid \mathcal{A}^{-1}(a), a) \cdot \mathbb{P}^{d^\infty}\{Z_t = i\}, \text{ for } \beta(a) = i, \beta(s) = j \\ &= \mathbb{P}^{d^\infty}\{Z_t = i\} \cdot \sum_{j:(i,j) \in E} p(s \mid \mathcal{A}^{-1}(a), a), \text{ for } \beta(a) = i, \beta(s) = j \\ &= \mathbb{P}^{d^\infty}\{Z_t = i\}. \end{aligned}$$

For the second statement, a similar proof is possible. Let $t \in \mathbb{N}$, $t \geq 2$. Then,

$$\sum_{j:(j,i) \in E} x_{j,i}^{t-1} \stackrel{\text{Def.3.6.9}}{=} \sum_{j:(j,i) \in E} \mathbb{P}^{d^\infty}\{Z_t = i, Z_{t-1} = j\}.$$

Again, for $\mathbb{P}^{d^\infty}\{Z_t = i\} = 0$, we have that $\mathbb{P}^{d^\infty}\{Z_t = i, Z_{t-1} = j\} = 0$ for all $(j, i) \in E$ and the statement holds. Otherwise, due to the definition of E , there must exist at least one j with $(j, i) \in E$ and $\mathbb{P}^{d^\infty}\{Z_{t-1} = j\} > 0$. Hence, it holds

$$\sum_{j:(j,i) \in E} \mathbb{P}^{d^\infty}\{Z_t = i, Z_{t-1} = j\} = \sum_{\substack{j:(j,i) \in E \\ \mathbb{P}^{d^\infty}\{Z_{t-1}=j\}>0}} \mathbb{P}^{d^\infty}\{Z_t = i \mid Z_{t-1} = j\} \cdot \mathbb{P}^{d^\infty}\{Z_{t-1} = j\}.$$

This time for $i \in N_S$, it holds

$$\mathbb{P}^{d^\infty} \{Z_t = i \mid Z_{t-1} = j\} = p(s|A^{-1}(a), a), \text{ for } \beta(a) = j, \beta(s) = i$$

and hence

$$\begin{aligned} & \sum_{\substack{j:(i,j) \in E \\ \mathbb{P}^{d^\infty} \{Z_{t-1}=j\} > 0}} \mathbb{P}^{d^\infty} \{Z_t = i \mid Z_{t-1} = j\} \cdot \mathbb{P}^{d^\infty} \{Z_{t-1} = j\} \\ = & \sum_{\substack{j:(i,j) \in E \\ \mathbb{P}^{d^\infty} \{Z_{t-1}=j\} > 0}} p(s|A^{-1}(a), a) \cdot \mathbb{P}^{d^\infty} \{Z_{t-1} = j\}, \text{ for } \beta(a) = j, \beta(s) = i \\ = & \mathbb{P}^{d^\infty} \{Z_t = i\}. \end{aligned}$$

In the last equality, it was used that for each action $\beta^{-1}(j)$ that has a positive probability for a transition to $\beta^{-1}(i)$ an edge $(j, i) \in E$ exists. For $i \in N_A$

$$\mathbb{P}^{d^\infty} \{Z_t = i \mid Z_{t-1} = j\} = q_{d(s)}(a), \text{ for } \beta(s) = j, \beta(a) = i$$

and the same argumentation applies. ■

Theorem 3.6.11 is a direct application of the two statements of the last Proposition. It shows that $x_{i,j}^t$ derived from Z_t after Definition 3.6.9 is a valid flow in each node at time $t \geq 2$. The theorem holds only for $t \geq 2$ since $\{x_{i,j}^0, (i, j) \in E\}$ is not defined.

Theorem 3.6.11 (Time varying flow induced by Z_t meets flow condition):

Let $x_{i,j}^t$ be the induced time-dependent flow by Z_t defined according to definition 3.6.9. Then $x_{i,j}^t$ is a valid flow in the time-expanded flow network with adapted edge weights, i.e.,

$$\sum_{j:(i,j) \in E} x_{i,j}^t - \sum_{j:(j,i) \in E} x_{j,i}^{t-1} = 0, \quad \forall t \geq 2, \quad \forall i \in N.$$

We have seen that the time depending flow variables induced by Z_t satisfy the flow condition in the time-expanded flow network. As the name suggest, the flow on one edge is not static and may change over time. In our Example of Figure 3.6, the time varying flows on the edge (1, 2) are:

$$\begin{aligned} x_{(1,2)}^1 &= 1 \\ x_{(1,2)}^2 &= 0 \\ x_{(1,2)}^3 &= 0.6 \\ x_{(1,2)}^4 &= 0 \dots \end{aligned}$$

However, the reader may observe that the fraction of flow available in a node that traverses an edge is static. In our example, if there is an incoming flow to node 1, the fraction of outgoing flow that traverses the edge (1, 2) is always 1 which equals d_{s_1} .

The static fraction always equals the adapted edge weight α_{d^∞} , which is, depending on the edge type, either equal to the probability of a transition or to the probability by which an action is chosen under the considered decision rule. This formally is shown by the next two theorems, whose proofs are based on our previous propositions:

Theorem 3.6.12 (Relation induced flow to decision rule):

Assume an SSO-MDP and a stationary policy d^∞ . Let $(N, E, \beta, \alpha_{d^\infty})$ be an associated flow network of that SSO-MDP and Z_t be the random variable of Definition 3.6.4. Then, for any node $i \in N_S$

$$\frac{x_{i,k}^t}{\sum_{j:(j,i) \in E} x_{j,i}^{t-1}} = q_{d(\beta^{-1}(i))}(\beta^{-1}(k)), \quad \forall k : (i, k) \in E$$

for all $t \geq 2$ with $\sum_{j:(j,i) \in E} x_{j,i}^{t-1} > 0$.

PROOF. According to Proposition 3.6.10

$$\sum_{j:(j,i) \in E} x_{j,i}^{t-1} = \mathbb{P}^{d^\infty} \{Z_t = i\}.$$

For $\mathbb{P}^{d^\infty} \{Z_t = i\} > 0$ the following equation holds:

$$\begin{aligned} \frac{x_{i,k}^t}{\sum_{j:(j,i) \in E} x_{j,i}^{t-1}} &= \frac{\mathbb{P}^{d^\infty} \{Z_{t+1} = k, Z_t = i\}}{\mathbb{P}^{d^\infty} \{Z_t = i\}} \\ &= \mathbb{P}^{d^\infty} \{Z_{t+1} = k \mid Z_t = i\} \\ &\stackrel{\text{Prop. 3.6.8}}{=} \alpha_{d^\infty}(i, k) \\ &= q_{d(\beta^{-1}(i))}(\beta^{-1}(k)). \end{aligned} \tag{3.7}$$

In Equation 3.7, Proposition 3.6.8 is used and the property of the time-expanded graph that there exists only one path from (t, i) to $(t+1, k)$, which equals the direct edge $(t, i) - (t+1, k)$. ■

A similar relation holds for action nodes $i \in N_A$ as shown in Theorem 3.6.13. For a shorter notation, $p(s|a) := p(s|\mathcal{A}^{-1}(a), a)$ is used for the transition probability from action $a \in \mathcal{A}$ to state $s \in S$.

Theorem 3.6.13 (Relation transition probabilities induced flow):

Assume an SSO-MDP and a stationary policy d^∞ . Let $(N, E, \beta, \alpha_{d^\infty})$ be an associated flow network of that SSO-MDP and Z_t be the random variable of Definition 3.6.4. Then, for any node $i \in N_A$

$$\frac{x_{i,k}^t}{\sum_{j:(j,i) \in E} x_{j,i}^{t-1}} = p(\beta^{-1}(k) | \beta^{-1}(i)), \forall k : (i, k) \in E$$

for all $t \geq 2$ with $\sum_{j:(j,i) \in E} x_{j,i}^{t-1} > 0$.

PROOF. Again, according to Proposition 3.6.10

$$\sum_{j:(j,i) \in E} x_{j,i}^{t-1} = \mathbb{P}^{d^\infty} \{Z_t = i\}.$$

For $\mathbb{P}^{d^\infty} \{Z_t = i\} > 0$ the following equation holds:

$$\begin{aligned} \frac{x_{i,k}^t}{\sum_{j:(j,i) \in E} x_{j,i}^{t-1}} &= \frac{\mathbb{P}^{d^\infty} \{Z_{t+1} = k, Z_t = i\}}{\mathbb{P}^{d^\infty} \{Z_t = i\}} \\ &= \mathbb{P}^{d^\infty} \{Z_{t+1} = k \mid Z_t = i\} \\ &\stackrel{\text{Prop. 3.6.8}}{=} \alpha_{d^\infty}(i, k) \\ &= p(\beta^{-1}(k) | \beta^{-1}(i)). \end{aligned} \tag{3.8}$$

In Equation 3.8, Proposition 3.6.8 is used again as well as the property of the time-expanded graph that there exists only one path from (t, i) to $(t + 1, k)$, which equals the direct edge $(t, i) - (t + 1, k)$. In contrast to the last theorem, $(t, i) - (t + 1, k)$ is an edge from an action node to a state node, so $\alpha_{d^\infty}(i, k)$ equals the transition probability $p(\beta^{-1}(k) | \beta^{-1}(i))$. ■

The last two theorems capture all nodes N in the flow network, since all nodes belong either to N_S or to N_A . As we are in a setting with stationary transition probabilities and consider only stationary policies, it can be followed from the last two theorems that the fraction of the available flow in a node that traverses an edge (i, j) is constant over time.

Corollary 3.6.14:

Assume an SSO-MDP and a stationary policy d^∞ . Let $(N, E, \beta, \alpha_{d^\infty})$ be an associated flow network of that SSO-MDP and Z_t be the random variable of Definition 3.6.4. Then for nodes i with $\sum_{k:(k,i) \in E} x_{k,i}^{t-1} > 0$ and $\sum_{k:(k,i) \in E} x_{k,i}^{t'-1} > 0$ the equation

$$\frac{x_{i,j}^t}{\sum_{k:(k,i) \in E} x_{k,i}^{t-1}} = \frac{x_{i,j}^{t'}}{\sum_{k:(k,i) \in E} x_{k,i}^{t'-1}}$$

holds for all $t, t' \geq 2$.

In the case $\sum_{k:(k,i) \in E} x_{k,i}^t = 0$, the incoming flow to node i at time t is zero, which means, that the probability that Z_t is at time t in node i is zero. Since x^t satisfies the flow condition, it follows that all outgoing flows of node i at time $t + 1$ must be zero, too. So, $x_{i,j}^{t+1} = 0$ for all $(i, j) \in E$.

It may be the case that $\sum_{k:(k,i) \in E} x_{k,i}^t = 0$ holds for some t but not for all. Consider for example node 2 in Example 3.4. The incoming flow at time $t = 2$ is

$$\sum_{k:(k,2) \in E} x_{k,2}^2 = 1$$

while at time $t = 3$ this sum is 0. However, for all points in time where the incoming flow into a node is positive, the fraction of flow on an outgoing edge is always the same. Nodes which never have a positive incoming flow are not visited by Z_t under the considered decision rule.

In the current subsection, the properties of the induced flow of the random variable Z_t are analyzed for a fixed stationary policy d^∞ . In the next subsection, this view will be inverted. Coming from a valid flow in the flow network of an SSO-MDP a stationary policy d^∞ is derived. It is shown that the random variable Z_t under d^∞ induces exactly the flow from which it was derived.

3.6.3 Maximum Flow Problem for SSO-MDPs

In this section, a linear programming formulation for SSO-MDPs is derived by considering the associated flow network of an SSO-MDP. The maximum flow problem formulation of Definition 2.4.8 is adapted to flow networks of SSO-MDPs such that for each feasible solution of the linear program, a decision rule can be found under which Z_t 's probability of reaching a winning state equals the objective value of that solution.

The objective of an SSO-MDP is to find an optimal stationary policy d^∞ such that the probability of reaching a winning state is maximized, which is shown in Proposition 3.4.1. In the time-expanded flow network of an associated graph of that SSO-MDP, this objective translates to maximizing the probability that the random variable Z_t enters a winning node of the node set N_S^W , see Proposition 3.6.6. In the last subsection in Proposition 3.6.11, we have seen that the probability distribution of Z_t induces a valid flow in the time-expanded network. This leads to the idea of formulating a maximum flow problem for SSO-MDPs. In the maximum flow problem formulation, the evolution of Z_t is considered until it reaches a winning state. Therefore, in this section an associated graph of an SSO-MDP is used where the node set does not include the artificial action nodes at the absorbing states. This definition is important such that the resulting linear program will be well defined. So, in this section an associated *linear programming* graph $G^{LP} = (\tilde{N}, \tilde{E}, \tilde{\beta})$ of an SSO-MDP is considered. Its sets are defined as follows:

$$\begin{aligned} \tilde{N} &:= \{1, \dots, |S \cup A \setminus \cup_{s \in W \cup L} A_s|\} \\ \tilde{E} &:= \{(i, j) \in E, i, j \in \tilde{N}\} \end{aligned}$$

and $\tilde{\beta}$ is a bijection from $S \cup A \setminus \{ \cup_{s \in W \cup L} A_s \}$ to \tilde{N} . The associated flow network based on an associated linear programming graph G^{LP} is denoted by $(\tilde{N}, \tilde{E}, \tilde{\beta}, \alpha)$ and α is defined according to Definition 3.6.3 but based on G^{LP} . Similar, the time-extended flow network of an SSO-MDP can be constructed according to Definition 3.6.5 based on an associated *linear programming* graph G^{LP} and is denoted by $(\tilde{N}_t, \tilde{E}_t, \tilde{\beta}_t, \alpha^t)$.

The meaning of the objective function of an SSO-MDP and the properties of Z_t in the time-expanded network lead to the idea of using a maximum flow formulation as a starting point for the linear program. The basic max flow formulation of Subsection 2.4 for a capacitated network with a non-negative capacity u_{ij} based on a graph $G = (N, E)$ is:

$$\left| \begin{array}{l} \max v \\ \sum_{j:(i,j) \in E} x_{ij} - \sum_{j:(j,i) \in E} x_{ji} = \begin{cases} v & \text{for } i = s \\ 0 & \text{for all } i \in N \setminus \{s, t\} \\ -v & \text{for } i = t \end{cases} \\ 0 \leq x_{ij} \leq u_{ij}, \quad \forall (i, j) \in E. \end{array} \right| \quad (\text{max flow})$$

There are obviously some general differences between the network considered in Definition 2.4.7 and the time-expanded network $(\tilde{N}^t, \tilde{E}^t, \tilde{\beta}^t, \alpha^t)$ of an SSO-MDP.

First, there are no capacities on the edges \tilde{E}^t . If the starting flow v is increased from the source node, we would be able to route the full amount v over the paths in the network. And if there exists a path to a winning node, the objective function would be unbounded. However, the flow in the time-expanded network should equal the flow induced by the random variable Z_t . So, it makes sense to fix the amount of flow v entering the network from the outside to 1. In fact, the initial distribution P_1 of the starting state is used to model the incoming flow to the network. Even if in SSO-MDPs the starting state s_1 is known, in this subsection a generalized initial distribution P_1 for the starting state is used. The initial distribution P_1 always satisfies

$$\sum_{s \in \mathcal{S}} P_1(s) = \sum_{n \in \mathcal{N}_s} P_1(\tilde{\beta}^{-1}(n)) = 1.$$

For not having problems with the domain space of P_1 , P_1 is extended to the action space and $P_1(a) = 0$ for all $a \in \mathcal{A}$. For modeling the incoming flow, the edge set \tilde{E}^t of the time-expanded flow network of an SSO-MDP is extended by edges $\tilde{E}^0 := \{(0, s), (1, i)\}$, $i \in \mathcal{N}_s$ that connect an artificial source node s with each state node of the time-expanded flow network at time 0.

Instead of capacities, there exist edge weights α^t in a time-expanded flow network of an SSO-MDP. As seen in Proposition 3.6.8, the adapted edge weight $\alpha_{d^\infty}^t$ of an edge $(t, i) - (t + 1, j)$ equals the probability $\mathbb{P}^{d^\infty} \{Z_{t+1} = j \mid Z_t = i\}$. Since the flow in the network should correspond to the probability distribution of the random variable Z_t , further equalities are added to the program that ensure that the flow available in a node is distributed by the edge weights over the outgoing edges. Since we are searching for a decision rule d , the adapted edge weights $\alpha_{d^\infty}^t$ can not be used. Instead, variables $\delta(j)$ are defined for the distribution of the probability mass along edges that go from a state node to an action node.

Furthermore, there are several sink nodes $\mathbb{N} \times \{\mathcal{N}_s^W \cup \mathcal{N}_s^L\} \subseteq \tilde{N}_t$. A flow value variable v_i^t is used for each node $i \in \mathcal{N}_s^W \cup \mathcal{N}_s^L$ to capture the incoming flow in a winning or losing node at each point in time.

The program resulting from this considerations on basis of the maximum flow problem is presented in Definition 3.6.15. It has a countable infinite number of variables and the objective function is an infinite series over time that sums up the amount of flow entering a winning node at each point in time. Since the total flow entering the network is bounded by 1 through the initial distribution P_1 and no

additional flow can be generated, the objective function is bounded by 1. Therefore, the series must converge to a finite value.

Definition 3.6.15 (Time expanded Maximum Flow Problem for SSO-MDPs).

The time-expanded maximum flow problem of the time-expanded flow network $(\tilde{N}^t, \tilde{E}^t, \tilde{\beta}^t, \alpha^t)$ of an SSO-MDP and an initial distribution P_1 of the starting state is defined as

$$\left(\begin{array}{ll} \max \sum_{t \in \mathbb{N}} \sum_{i \in N_S^W} v_i^t & \\ \sum_{j:(j,i) \in \tilde{E}_N^0} x_{j,i}^0 = P_1(\tilde{\beta}^{-1}(i)), \quad \forall i \in \tilde{N}, & \\ \sum_{j:(i,j) \in \tilde{E}_N^t} x_{i,j}^t - \sum_{j:(j,i) \in \tilde{E}_N^{t-1}} x_{j,i}^{t-1} = 0, \quad \forall i \in \tilde{N} \setminus \{N_S^W \cup N_S^L\}, t \in \mathbb{N}, & \\ - \sum_{j:(j,i) \in \tilde{E}_N^{t-1}} x_{j,i}^{t-1} = -v_i^t, \quad \forall i \in N_S^W \cup N_S^L, t \in \mathbb{N}, & \\ x_{i,j}^t - \alpha(i,j) \sum_{k:(k,i) \in \tilde{E}_N^{t-1}} x_{k,i}^{t-1} = 0, \quad \forall (i,j) \in \tilde{E}_N^t, i \in \tilde{N}_A, t \in \mathbb{N}, & \\ x_{i,j}^t - \delta(j) \sum_{k:(k,i) \in \tilde{E}_N^{t-1}} x_{k,i}^{t-1} = 0, \quad \forall (i,j) \in \tilde{E}_N^t, i \in N_S, t \in \mathbb{N}, & \\ x_{i,j}^t \geq 0, \quad \forall (i,j) \in \tilde{E}_N^t, t \in \mathbb{N}_0, & \end{array} \right) \quad (3.9)$$

where $x_{i,j}^t, \delta(j)$ as well as v_i^t are variables. Furthermore, \tilde{E}^t is extended by $\tilde{E}^0 := \{((0, s), (1, i)), i \in N_S\}$ and \tilde{E}_N^t is defined as

$$\tilde{E}_N^t := \{(i, j) \mid ((t, i), (t+1, j)) \in \tilde{E}_t\}, \forall t \in \mathbb{N}_0.$$

In the following, the first three sets of equalities in the program are referenced as *flow constraints* and the last two sets of equalities as *flow distribution constraints*.

For easier notation, \tilde{E}_N^t is used for all $t \in \mathbb{N}_0$ which is defined as a projection of the time-expanded edges at time t on edges between the static nodes $i \in \tilde{N}$. Since an SSO-MDPs has stationary parameters, for all $t \in \mathbb{N}$, which does not include 0, it holds that \tilde{N}_t restricted to the node component equals \tilde{N} , \tilde{E}_N^t equals \tilde{E} and $\alpha^t(i, j) = \alpha(i, j)$.

The flow distribution constraints for state nodes are non-linear which is why the whole program is not a linear program. The non-linear flow distribution constraints are needed to ensure a stationary decision rule in the time expanded setting. However, in the static reformulation of this program the non-linear constraint can be omitted such that the static reformulation is a linear program.

From every feasible solution of problem 3.9, a decision rule d can be defined such that the induced flow of Z_t under d^∞ equals the feasible solution $x_{i,j}^t$. This is shown in the next theorem:

Theorem 3.6.16 (Feasible solution of 3.9 characterizes decision rule):

Assume the time-extended flow network $(\tilde{N}^t, \tilde{E}^t, \tilde{\beta}^t, \alpha^t)$ of an associated graph of an SSO-MDP and an initial distribution P_1 of the starting state. Let $x_{i,j}^t, v_i^t$ and $\delta(j)$ be a feasible solution of

program 3.9 defined for that time-expanded network. Define a decision rule $d \in D^{MR}$ as

$$\mathbb{P}\{d(s) = a\} := \begin{cases} \delta(j) & \text{for all } i \in N_S \text{ with } \exists t \in \mathbb{N} \text{ s.t.} \\ & \sum_{k:(k,i) \in \tilde{E}_N^{t-1}} x_{k,i}^{t-1} > 0, \\ q_s(a) \text{ arbitrary s.t.} & \\ \sum_{a \in A_s} q_s(a) = 1, & \text{else} \\ q_s(a) \geq 0, & \end{cases}$$

for all $a \in A_s$, where $j = \tilde{\beta}(a)$ and $i = \tilde{\beta}(s)$. Then,

1. the decision rule $d \in D^{MR}$ is well-defined,
2. $\mathbb{P}^{d^\infty}\{Z_1 = i\} = \sum_{k:(i,k) \in \tilde{E}} x_{i,k}^1$
3. $\mathbb{P}^{d^\infty}\{Z_{t+1} = j, Z_t = i\} = x_{i,j}^t, \forall (i,j) \in \tilde{E}, t \in \mathbb{N}$.

PROOF. Let $x_{i,j}^t, v_i^t$ and $\delta(j)$ be a feasible solution of program 3.9. Define a decision rule $d \in D^{MR}$ as described in the theorem. Remember, for SSO-MDPs $\tilde{E}_N^t = \tilde{E}$ for all $t \in \mathbb{N}$.

1. First, it is shown that d is well-defined. For every $s \in S$, the decision rule d must specify a probability distribution over the available actions in s . This means $s = \tilde{\beta}^{-1}(i)$ with

$$\exists t \in \mathbb{N} : \sum_{k:(k,i) \in \tilde{E}_N^{t-1}} x_{k,i}^{t-1} > 0.$$

δ must satisfy:

$$\sum_{a \in A_s} \delta(\tilde{\beta}(a)) = 1 \quad \wedge \quad \delta(\tilde{\beta}(a)) \geq 0, \forall a \in A_s, s \in S$$

such that d is well-defined. From the flow distribution equation of the program for state nodes, it follows

$$\delta(j) = \frac{x_{i,j}^t}{\sum_{k:(k,i) \in \tilde{E}_N^{t-1}} x_{k,i}^{t-1}}.$$

After Corollary 3.6.14, any time point t with $\sum_{k:(k,i) \in \tilde{E}_N^{t-1}} x_{k,i}^{t-1} > 0$ yields to the same $\delta(j)$. The non-negativity of δ follows directly from the non-negativity of $x_{i,j}^t$.

By the definition of the edges in the flow network of an associated graph of an SSO-MDP, the following equation holds:

$$\sum_{a \in A_s} \delta(\tilde{\beta}(a)) = \sum_{j:(i,j) \in \tilde{E}_N^t} \delta(j) = \sum_{j:(i,j) \in \tilde{E}} \delta(j), \forall t \in \mathbb{N}.$$

Using that $x_{i,j}^t$ satisfies the flow condition, it holds for i :

$$\begin{aligned} \sum_{j:(i,j) \in \tilde{E}} \delta(j) &= \sum_{j:(i,j) \in \tilde{E}} \left(\frac{x_{i,j}^t}{\sum_{k:(k,i) \in \tilde{E}^{t-1}} x_{k,i}^{t-1}} \right) \\ &= \sum_{j:(i,j) \in \tilde{E}} \left(\frac{x_{i,j}^t}{\sum_{k:(i,k) \in \tilde{E}^t} x_{i,k}^t} \right) \\ &= \frac{\sum_{j:(i,j) \in \tilde{E}} x_{i,j}^t}{\sum_{k:(i,k) \in \tilde{E}} x_{i,k}^t} \\ &= 1. \end{aligned}$$

If there exists no t such that the incoming flow of a state node i greater zero, the decision rule d is set to an arbitrary probability distribution over the available actions in $\tilde{\beta}^{-1}(i)$. So, d is well defined in any state s .

It is now proven that Z_t under d induces a flow that is equivalent to $x_{i,j}^t$:

- Z_1 is distributed according to the initial start distribution P_1 . Due to the first inequality in the program the following equation holds:

$$\mathbb{P}^{d^\infty} \{Z_1 = i\} = P_1(\tilde{\beta}^{-1}(i)) \stackrel{\text{Initial Cons.}}{=} \sum_{k:(k,i) \in \tilde{E}_N^0} x_{k,i}^0 \stackrel{\text{Flow Cons.}}{=} \sum_{k:(i,k) \in \tilde{E}_N^1} x_{i,k}^1 = \sum_{k:(i,k) \in \tilde{E}} x_{i,k}^1.$$

- By induction, it is shown that $\mathbb{P}^{d^\infty} \{Z_{t+1} = j, Z_t = i\} = x_{i,j}^t, \forall (i,j) \in \tilde{E}, t \in \mathbb{N}$.

Induction start: Let $t = 1$.

Case $\sum_{k:(k,i) \in \tilde{E}_N^0} x_{k,i}^0 > 0$: Then due to the initial constraint, it holds that $P_1(\tilde{\beta}^{-1}(i)) > 0$ and hence i must be a state node. From the definition of d it follows

$$\begin{aligned} \mathbb{P}^{d^\infty} \{Z_2 = j, Z_1 = i\} &= P_1(\tilde{\beta}^{-1}(i)) \cdot \mathbb{P}^{d^\infty} \{d(\tilde{\beta}^{-1}(i)) = \tilde{\beta}^{-1}(j)\} \\ &\stackrel{\text{initial Cons.}}{=} \left(\sum_{k:(k,i) \in \tilde{E}_N^0} x_{k,i}^0 \right) \cdot \delta(j) \\ &\stackrel{\text{flow distr. Cons.}}{=} x_{i,j}^1. \end{aligned}$$

The last equality follows from the flow distribution equality for state nodes in the program.

Case $\sum_{k:(k,i) \in \tilde{E}_N^0} x_{k,i}^0 = 0$: In this case, $\mathbb{P}^{d^\infty} \{Z_1 = i\} = 0$ because

$$0 = \sum_{k:(k,i) \in \tilde{E}_N^0} x_{k,i}^0 = P_1(\tilde{\beta}^{-1}(i)) = \mathbb{P}^{d^\infty} \{Z_1 = i\}.$$

From $\mathbb{P}^{d^\infty} \{Z_1 = i\} = 0$, it can be concluded that

$$\mathbb{P}^{d^\infty} \{Z_2 = j, Z_1 = i\} = 0$$

for all $j \in \tilde{N}$, $(i, j) \in \tilde{E}$. So, it has to be shown that $x_{i,j}^1$ is zero for all $j \in \tilde{N}$, $(i, j) \in \tilde{E}$. This is obviously satisfied because of the flow distribution constraints: If i is a state node, it holds

$$0 = \delta(j) \underbrace{\sum_{k:(k,i) \in \tilde{E}_N^0} x_{k,i}^0}_{=0} = x_{i,j}^1, \quad \forall (i, j) \in \tilde{E}_N^1, \quad \forall \delta(j) \in [0, 1].$$

If i is an action node, it holds

$$0 = \alpha(i, j) \underbrace{\sum_{k:(k,i) \in \tilde{E}_N^0} x_{k,i}^0}_{=0} = x_{i,j}^1, \quad \forall (i, j) \in \tilde{E}_N^1, \quad \forall \alpha(i, j) \in [0, 1].$$

Induction Step: Assume for $t - 1 \in \mathbb{N}$ the induction hypothesis holds for all $(i, j) \in \tilde{E}$.

Case $\sum_{k:(k,i) \in \tilde{E}_N^{t-1}} x_{k,i}^{t-1} > 0$: If $i \in N_S$, it follows

$$\begin{aligned} \mathbb{P}^{d^\infty} \{Z_{t+1} = j, Z_t = i\} &= \mathbb{P}^{d^\infty} \{Z_{t+1} = j \mid Z_t = i\} \cdot \mathbb{P}^{d^\infty} \{Z_t = i\} \\ &= \mathbb{P}^{d^\infty} \{d(\tilde{\beta}^{-1}(i)) = \tilde{\beta}^{-1}(j)\} \cdot \sum_{k \in \tilde{N}_A} \mathbb{P}^{d^\infty} \{Z_t = i, Z_{t-1} = k\} \\ &\stackrel{(IH)}{=} \delta(j) \cdot \left(\sum_{k:(k,i) \in \tilde{E}} x_{k,i}^{t-1} \right) \\ &= x_{i,j}^t. \end{aligned}$$

If $i \in \tilde{N}_A$, it follows

$$\begin{aligned} \mathbb{P}^{d^\infty} \{Z_{t+1} = j, Z_t = i\} &= \mathbb{P}^{d^\infty} \{Z_{t+1} = j \mid Z_t = i\} \cdot \mathbb{P}^{d^\infty} \{Z_t = i\} \\ &= p(\tilde{\beta}^{-1}(j) \mid \tilde{\beta}^{-1}(i)) \cdot \sum_{k \in N_S} \mathbb{P}^{d^\infty} \{Z_t = i, Z_{t-1} = k\} \\ &\stackrel{(IH)}{=} \alpha(i, j) \cdot \left(\sum_{k:(k,i) \in \tilde{E}} x_{k,i}^{t-1} \right) \\ &= x_{i,j}^t. \end{aligned}$$

Case $\sum_{k:(k,i) \in \tilde{E}_N^{t-1}} x_{k,i}^{t-1} = 0$: The probability $\mathbb{P}^{d^\infty} \{Z_t = i\} = 0$, since

$$0 = \sum_{k:(k,i) \in \tilde{E}_N^{t-1}} x_{k,i}^{t-1} \stackrel{(IH)}{=} \sum_{k:(k,i) \in \tilde{E}} \mathbb{P}^{d^\infty} \{Z_t = i, Z_{t-1} = k\}.$$

From $\mathbb{P}^{d^\infty} \{Z_t = i\} = 0$, it can be derived that

$$\mathbb{P}^{d^\infty} \{Z_{t+1} = j, Z_t = i\} = 0$$

for all $j \in \tilde{N}$, $(i, j) \in \tilde{E}$. So, it has to be shown that $x_{i,j}^t$ is zero for all $j \in \tilde{N}$, $(i, j) \in \tilde{E}$. This is obviously satisfied because of the flow distribution constraints: If i is a state node, it holds

$$0 = \delta(j) \underbrace{\sum_{k:(k,i) \in \tilde{E}} x_{k,i}^{t-1}}_{=0} = x_{i,j}^t, \quad \forall (i, j) \in \tilde{E}, \quad \forall \delta(j) \in [0, 1].$$

If i is an action node, it holds

$$0 = \alpha(i, j) \underbrace{\sum_{k:(k,i) \in \tilde{E}} x_{k,i}^{t-1}}_{=0} = x_{i,j}^t, \quad \forall (i, j) \in \tilde{E}, \quad \forall \alpha(i, j) \in [0, 1].$$

■

From the last theorem and the characterization of the value of a policy in an SSO-MDP, it can be concluded that the objective value of a feasible solution of program 3.9 corresponds to the value of the derived stationary policy in the SSO-MDP:

Corollary 3.6.17 (Objective value of a feasible solution):

Assume the time-extended flow network $(\tilde{N}^t, \tilde{E}^t, \tilde{\beta}^t, \alpha^t)$ of an associated graph of an SSO-MDP and an initial distribution P_1 of the starting state. Let $x_{i,j}^t, v_i^t$ and $\delta(j)$ be a feasible solution of program 3.9 defined for that time-expanded network. Define a decision rule $d \in D^{MR}$ as in Theorem 3.6.16. It can be concluded that the objective value of $x_{i,j}^t, v_i^t$ and $\delta(j)$ in program 3.9 equals the value of the stationary policy v^{d^∞} under the initial distribution P_1 :

$$\sum_{t \in \mathbb{N}} \sum_{i \in N_S^W} v_i^t = \sum_{s \in S \setminus W} v^{d^\infty}(s) \cdot P_1(s) + \sum_{s \in W} P_1(s).$$

PROOF. Let $x_{i,j}^t$ be a feasible solution of program 3.9. Then,

$$\begin{aligned} & \sum_{t \in \mathbb{N}} \sum_{i \in N_S^W} v_i^t \\ = & \sum_{t \in \mathbb{N}} \sum_{i \in N_S^W} \sum_{j:(j,i) \in \tilde{E}_N^{t-1}} x_{j,i}^{t-1} \\ = & \sum_{t \in \mathbb{N}} \sum_{i \in N_S^W} \sum_{j:(j,i) \in \tilde{E}} x_{j,i}^t + \sum_{i \in N_S^W} \sum_{j:(j,i) \in \tilde{E}_N^0} x_{j,i}^0 \\ \stackrel{\text{Thm. 3.6.16}}{=} & \sum_{t \in \mathbb{N}} \sum_{i \in N_S^W} \sum_{j:(j,i) \in \tilde{E}} \mathbb{P}^{d^\infty} \{Z_{t+1} = i, Z_t = j\} + \sum_{i \in N_S^W} P_1(\tilde{\beta}^{-1}(i)) \\ \stackrel{\text{Def. } (\tilde{N}^t, \tilde{E})}{=} & \sum_{t=2}^{\infty} \mathbb{P}^{d^\infty} \{Z_{t+1} \in N_S^W, Z_{t-1} \notin N_S^W\} + \sum_{i \in N_S^W} P_1(\tilde{\beta}^{-1}(i)) \end{aligned}$$

$$\begin{aligned}
& \stackrel{t \text{ odd}}{=} \sum_{t=2}^{\infty} \mathbb{P}^{d^{\infty}} \{Z_{2t-3} \notin N_S^W, Z_{2t-1} \in N_S^W\} + \sum_{i \in N_S^W} P_1(\tilde{\beta}^{-1}(i)) \\
& = \sum_{i \in \tilde{N} \setminus N_S^W : P_1(\tilde{\beta}^{-1}(i)) > 0} \left(\sum_{t=2}^{\infty} \mathbb{P}^{d^{\infty}} \{Z_{2t-3} \notin N_S^W, Z_{2t-1} \in N_S^W \mid Z_1 = i\} \cdot \mathbb{P}^{d^{\infty}} \{Z_1 = i\} \right) \\
& \quad + \sum_{i \in N_S^W} P_1(\tilde{\beta}^{-1}(i)) \\
& \stackrel{\text{Prop. 3.6.6}}{=} \sum_{s \in S \setminus W} v^{d^{\infty}}(s) \cdot P_1(s) + \sum_{i \in W} P_1(s).
\end{aligned}$$

■

Even if it is not possible to implement program 3.9 due to its infinite number of inequalities and variables, the formulation for the example problem is stated in the example below:

Example 3.5 (Time expanded formulation):

Consider the time-expanded graph of Figure 3.6, which is associated with the SSO-MDP of Example 3.2. Assume the initial distribution $P_1(\tilde{\beta}^{-1}(1)) = 1$. The time-expanded maximum flow formula-

tion for this example is:

$$\begin{array}{rcl}
 \max \sum_{t \in \mathbb{N}} v_4^t & & \\
 x_{s,1}^0 & = & 1, \\
 x_{s,2}^0 & = & 0, \\
 x_{s,3}^0 & = & 0, \\
 x_{s,4}^0 & = & 0, \\
 x_{1,2}^1 - x_{s,1}^0 & = & 0, \\
 x_{1,2}^t - x_{2,1}^{t-1} & = & 0, \quad \forall t \in \mathbb{N}_{\geq 2}, \\
 x_{2,1}^1 + x_{2,3}^1 + x_{2,4}^1 - x_{s,2}^0 & = & 0, \\
 x_{2,1}^t + x_{2,3}^t + x_{2,4}^t - x_{1,2}^{t-1} & = & 0, \quad \forall t \in \mathbb{N}_{\geq 2}, \\
 -x_{s,3}^0 & = & -v_3^1, \\
 -x_{2,3}^{t-1} & = & -v_3^t, \quad \forall t \in \mathbb{N}_{\geq 2}, \\
 -x_{s,4}^0 & = & -v_4^1, \\
 -x_{2,4}^{t-1} & = & -v_4^t, \quad \forall t \in \mathbb{N}_{\geq 2}, \\
 x_{2,1}^1 - 0.6 \cdot x_{s,2}^0 & = & 0, \\
 x_{2,1}^t - 0.6 \cdot x_{1,2}^{t-1} & = & 0, \quad \forall t \in \mathbb{N}_{\geq 2}, \\
 x_{2,3}^1 - 0.2 \cdot x_{s,2}^0 & = & 0, \\
 x_{2,3}^t - 0.2 \cdot x_{1,2}^{t-1} & = & 0, \quad \forall t \in \mathbb{N}_{\geq 2}, \\
 x_{2,4}^1 - 0.2 \cdot x_{s,2}^0 & = & 0, \\
 x_{2,4}^t - 0.2 \cdot x_{1,2}^{t-1} & = & 0, \quad \forall t \in \mathbb{N}_{\geq 2}, \\
 x_{1,2}^1 - \delta(2) \cdot x_{s,1}^0 & = & 0, \\
 x_{1,2}^t - \delta(2) \cdot x_{2,1}^{t-1} & = & 0, \quad \forall t \in \mathbb{N}_{\geq 2}, \\
 x_{s,2}^0, x_{s,1}^0, x_{s,3}^0, x_{s,4}^0 & \geq & 0, \\
 x_{1,2}^t, x_{2,1}^t, x_{2,3}^t, x_{2,4}^t & \geq & 0, \quad \forall t \in \mathbb{N}.
 \end{array} \tag{3.10}$$

*

The goal is now to find a formulation that is easier to handle than the time-expanded maximum flow formulation. First, a linear program is derived from the last formulation. After it has been refined and it will be proven that the linear program determines an optimal decision rule of the SSO-MDP.

Consider program 3.9 and sum for each $i \in \tilde{N} \setminus \{N_S^W \cup N_S^L\}$ over the initial distribution constraint and all flow constraints:

$$\sum_{j:(j,i) \in \tilde{E}_{\tilde{N}}^0} x_{j,i}^0 + \sum_{t \in \mathbb{N}} \left(\sum_{j:(i,j) \in \tilde{E}_{\tilde{N}}^t} x_{i,j}^t - \sum_{j:(j,i) \in \tilde{E}_{\tilde{N}}^{t-1}} x_{j,i}^{t-1} \right) = P_1(\tilde{\beta}^{-1}(i)).$$

The sum can be rearranged and $\tilde{E}_{\tilde{N}}^t = \tilde{E}$, $\forall t \in \mathbb{N}$ can be used such that

$$\sum_{j:(i,j) \in \tilde{E}} \sum_{t \in \mathbb{N}} x_{i,j}^t - \sum_{j:(j,i) \in \tilde{E}} \sum_{t \in \mathbb{N}} x_{j,i}^t = P_1(\tilde{\beta}^{-1}(i)).$$

Define $x_{i,j} := \sum_{t \in \mathbb{N}} x_{i,j}^t$ and replace $\sum_{t \in \mathbb{N}} x_{i,j}^t$ by $x_{i,j}$. I called the result *the aggregated flow constraint*

$$\sum_{j:(i,j) \in \tilde{E}} x_{i,j} - \sum_{j:(j,i) \in \tilde{E}} x_{j,i} = P_1(\tilde{\beta}^{-1}(i)), \forall i \in \tilde{N} \setminus \{N_S^W \cup N_S^L\}.$$

For the nodes $i \in N_S^W \cup N_S^L$, the sum over all flow constraints can be computed as

$$\begin{aligned} & \sum_{j:(j,i) \in \tilde{E}_N^0} x_{j,i}^0 + \sum_{t \in \mathbb{N}} \sum_{j:(j,i) \in \tilde{E}_N^{t-1}} x_{j,i}^{t-1} = P_1(\tilde{\beta}^{-1}(i)) + \sum_{t \in \mathbb{N}} v_i^t \\ \Leftrightarrow & 2 \cdot \sum_{j:(j,i) \in \tilde{E}_N^0} x_{j,i}^0 + \sum_{t \in \mathbb{N}} \sum_{j:(j,i) \in \tilde{E}} x_{j,i}^t = P_1(\tilde{\beta}^{-1}(i)) + \sum_{t \in \mathbb{N}} v_i^t \\ \Leftrightarrow & \sum_{t \in \mathbb{N}} \sum_{j:(j,i) \in \tilde{E}} x_{j,i}^t = \sum_{t \in \mathbb{N}} v_i^t - \sum_{j:(j,i) \in \tilde{E}_N^0} x_{j,i}^0 \\ \Leftrightarrow & \sum_{t \in \mathbb{N}} \sum_{j:(j,i) \in \tilde{E}} x_{j,i}^t = \sum_{t=2}^{\infty} v_i^t. \end{aligned}$$

If again $\sum_{t \in \mathbb{N}} x_{i,j}^t$ is replaced by $x_{i,j}$ and v_i is defined by $\sum_{t=2}^{\infty} v_i^t$, the resulting aggregated flow constraints of winning or losing nodes are

$$- \sum_{j:(j,i) \in \tilde{E}} x_{j,i} = -v_i, \forall i \in N_S^W \cup N_S^L.$$

Also, for each $i \in \tilde{N}$ the sum over all flow distribution equalities is:

$$\begin{aligned} & \sum_{t \in \mathbb{N}} x_{i,j}^t - \sum_{t \in \mathbb{N}} \left(\alpha(i,j) \sum_{k:(k,i) \in \tilde{E}_N^{t-1}} x_{k,i}^{t-1} \right) = 0, \forall (i,j) \in \tilde{E}, i \in \tilde{N}_A, \\ & \sum_{t \in \mathbb{N}} x_{i,j}^t - \sum_{t \in \mathbb{N}} \left(\delta(j) \sum_{k:(k,i) \in \tilde{E}_N^{t-1}} x_{k,i}^{t-1} \right) = 0, \forall (i,j) \in \tilde{E}, i \in N_S. \end{aligned}$$

Since $\alpha(i,j)$ and $\delta(j)$ are stationary, they can be excluded from the sum and $\sum_{t \in \mathbb{N}} x_{i,j}^t$ can be replaced by $x_{i,j}$:

$$\begin{aligned} & x_{i,j} - \alpha(i,j) \left(\sum_{k:(k,i) \in \tilde{E}} x_{k,i} + P_1(\tilde{\beta}^{-1}(i)) \right) = 0, \forall (i,j) \in \tilde{E}, i \in \tilde{N}_A, \\ & x_{i,j} - \delta(j) \left(\sum_{k:(k,i) \in \tilde{E}} x_{k,i} + P_1(\tilde{\beta}^{-1}(i)) \right) = 0, \forall (i,j) \in \tilde{E}, i \in N_S. \end{aligned}$$

The resulting program is

$$\left| \begin{array}{l} \max \sum_{i \in N_S^W} v_i + \sum_{i \in N_S^W} P_1(\tilde{\beta}^{-1}(i)) \\ \sum_{j:(i,j) \in \tilde{E}} x_{i,j} - \sum_{j:(j,i) \in \tilde{E}} x_{j,i} = P_1(\tilde{\beta}^{-1}(i)), \forall i \in \tilde{N} \setminus \{N_S^W \cup N_S^L\}, \\ \quad - \sum_{j:(j,i) \in \tilde{E}} x_{j,i} = -v_i, \forall i \in N_S^W \cup N_S^L, \\ x_{i,j} - \alpha(i,j) \left(\sum_{k:(k,i) \in \tilde{E}} x_{k,i} + P_1(\tilde{\beta}^{-1}(i)) \right) = 0, \forall (i,j) \in \tilde{E}, i \in \tilde{N}_A \\ x_{i,j} - \delta(j) \left(\sum_{k:(k,i) \in \tilde{E}} x_{k,i} + P_1(\tilde{\beta}^{-1}(i)) \right) = 0, \forall (i,j) \in \tilde{E}, i \in N_S \\ x_{i,j} \geq 0, \forall (i,j) \in \tilde{E}. \end{array} \right| \quad (3.11)$$

This program can be simplified by deleting redundant variables and constraints. After having developed the final program, the relation between both programs will be investigated in detail.

First, for all $i \in \tilde{N}_A$ the initial distribution $P_1(\tilde{\beta}^{-1}(i))$ is zero, because P_1 is a distribution over states. Furthermore, since the edge weights α are set to the transition probabilities, they define a probability distribution over the outgoing edges of an action node. Therefore, the flow condition in an action node is always satisfied:

$$\begin{aligned} & \sum_{j:(i,j) \in \tilde{E}} x_{i,j} - \sum_{j:(j,i) \in \tilde{E}} x_{j,i} \\ &= \sum_{j:(i,j) \in \tilde{E}} \alpha(i,j) \left(\sum_{k:(k,i) \in \tilde{E}} x_{k,i} + P_1(\tilde{\beta}^{-1}(i)) \right) - \sum_{j:(j,i) \in \tilde{E}} x_{j,i} \\ &= \sum_{k:(k,i) \in \tilde{E}} x_{k,i} + P_1(\tilde{\beta}^{-1}(i)) - \sum_{j:(j,i) \in \tilde{E}} x_{j,i} \\ &= P_1(\tilde{\beta}^{-1}(i)), \forall i \in \tilde{N}_A. \end{aligned}$$

Furthermore, this program can be simplified by inserting $x_{j,i} = \alpha(j,i) \sum_{k:(k,j) \in \tilde{E}} x_{k,j}$ for $j \in \tilde{N}_A$ in the flow conditions at a state node i . After this transformation all $x_{i,j}$ with $i \in \tilde{N}_A$ are eliminated in the program and these variables are not needed any more.

Also, the variables $\delta(j)$ and the corresponding equation can be removed from the program. This makes the aggregated program a linear program. In the time-expanded version, $\delta(j)$ was necessary to ensure a stationary policy. In the static program, there is no time dependence any more. Because of the flow condition, we can compute from any partition of the incoming flow in a state node a stationary decision rule. It will soon be established that a policy directly determined from $x_{i,j}$ satisfies our needs.

Because $P_1(\tilde{\beta}^{-1}(i))$ is a parameter of the SSO-MDP and not a variable, this constant term can be removed from the objective function. The resulting optimal solution will not change. However, it must be kept in mind that the objective value of the optimal solution in the refined program differs by this constant from the original value.

It would also be possible to eliminate the v_i for $i \in N_S^L$ and the corresponding equations. However, it is an easy check of the parameters if we can convince ourselves that for a feasible solution $\sum_{i \in N_S^W \cup N_S^L} v_i = 1$ holds. The refined static linear program is:

Definition 3.6.18 (Static Maximum Flow Problem for SSO-MDPS).

The static maximum flow problem of a flow network $(\tilde{N}, \tilde{E}, \tilde{\beta}, \alpha)$ of an SSO-MDP and an initial distribution P_1 of the starting state is defined as

$$\left| \begin{array}{l} \max \sum_{i \in N_S^W} v_i \\ \sum_{j:(i,j) \in \tilde{E}} x_{i,j} - \sum_{j:(j,i) \in \tilde{E}} \alpha(j,i) \sum_{k:(k,j) \in \tilde{E}} x_{k,j} = P_1(\tilde{\beta}^{-1}(i)), \forall i \in N_S \setminus \{N_S^W \cup N_S^L\} \\ - \sum_{j:(j,i) \in \tilde{E}} \alpha(j,i) \sum_{k:(k,j) \in \tilde{E}} x_{k,j} = -v_i, \forall i \in N_S^W \cup N_S^L \\ x_{i,j} \geq 0, \forall (i,j) \in \tilde{E}, i \in N_S. \end{array} \right| \quad (3.12)$$

In the following, it will be examined how the refined static maximum flow problem 3.12 is related to the time-dependent maximum flow problem 3.9. Even if the static maximum flow problem 3.12 is derived by equivalence transformations from the time-dependent maximum flow problem 3.9, it should be shown that for every feasible solution of the time-expanded program there exists a feasible solution of the static program.

Proposition 3.6.19:

Assume the flow network $(\tilde{N}, \tilde{E}, \tilde{\beta}, \alpha)$ of an associated graph of an SSO-MDP and an initial distribution P_1 of the starting state. Let

$$\begin{aligned} & \{x_{i,j}^t, (i,j) \in \tilde{E}_{\tilde{N}}^t, t \in \mathbb{N}_0\}, \\ & \{\delta(j), j \in \tilde{N}_A\}, \quad \text{be feasible for 3.9,} \\ & \{v_i^t, i \in N_S^W \cup N_S^L, t \in \mathbb{N}\}, \end{aligned}$$

where 3.9 is defined for the time-expanded flow network of $(\tilde{N}, \tilde{E}, \tilde{\beta}, \alpha)$. Then,

$$\left\{ \begin{array}{l} x_{i,j} := \sum_{t \in \mathbb{N}} x_{i,j}^t, (i,j) \in \tilde{E}, i \in N_S \\ v_i := \sum_{t=2}^{\infty} v_i^t, i \in N_S^W \cup N_S^L \end{array} \right\}, \quad \text{is feasible for 3.12}$$

defined for $(\tilde{N}, \tilde{E}, \tilde{\beta}, \alpha)$ and $\sum_{t \in \mathbb{N}} \sum_{i \in N_S^W} v_i^t = \sum_{i \in N_S^W} v_i + \sum_{i \in N_S^W} P_1(\tilde{\beta}^{-1}(i))$.

PROOF. Assume $x_{i,j}^t$, $\delta(j)$ and v_i^t are feasible for 3.9 and define $x_{i,j}$ and v_i as described in the Proposition.

It has to be shown that $x_{i,j}$ is feasible for the linear program 3.12 defined for $(\tilde{N}, \tilde{E}, \tilde{\beta}, \alpha)$. Since $x_{i,j}^t \geq 0$ holds, $x_{i,j}$ is also non-negative. The following equations show that $x_{i,j}$ satisfies the flow constraints for $i \in N_S \setminus \{N_S^W \cup N_S^L\}$:

$$\begin{aligned} & \sum_{j:(i,j) \in \tilde{E}} x_{i,j} - \sum_{j:(j,i) \in \tilde{E}} \alpha(j,i) \sum_{k:(k,j) \in \tilde{E}} x_{k,j} \\ \stackrel{\text{Def.}}{=} & \sum_{j:(i,j) \in \tilde{E}} \sum_{t \in \mathbb{N}} x_{i,j}^t - \sum_{j:(j,i) \in \tilde{E}} \alpha(j,i) \sum_{k:(k,j) \in \tilde{E}} \sum_{t \in \mathbb{N}} x_{k,j}^t \end{aligned}$$

$$\begin{aligned}
&= \sum_{j:(i,j) \in \tilde{E}} \sum_{t \in \mathbb{N}} x_{i,j}^t - \sum_{j:(j,i) \in \tilde{E}} \sum_{t \in \mathbb{N}} \alpha(j,i) \sum_{k:(k,j) \in \tilde{E}} x_{k,j}^t \\
\stackrel{\text{Flow distr.3.9}}{=} & \sum_{j:(i,j) \in \tilde{E}} \sum_{t \in \mathbb{N}} x_{i,j}^t - \sum_{j:(j,i) \in \tilde{E}} \sum_{t \in \mathbb{N}} x_{j,i}^{t+1} \\
&= \sum_{j:(i,j) \in \tilde{E}} \sum_{t=1}^{\infty} x_{i,j}^t - \sum_{j:(j,i) \in \tilde{E}} \sum_{t=2}^{\infty} x_{j,i}^t \\
&= \sum_{j:(i,j) \in \tilde{E}} x_{i,j}^1 + \underbrace{\sum_{j:(i,j) \in \tilde{E}} x_{i,j}^2}_{=0} + \sum_{j:(i,j) \in \tilde{E}} \sum_{t=3}^{\infty} x_{i,j}^t - \sum_{j:(j,i) \in \tilde{E}} \sum_{t=2}^{\infty} x_{j,i}^t \tag{3.13} \\
&= P_1(\tilde{\beta}^{-1}(i)) + \lim_{N \rightarrow \infty} \left(\sum_{t=3}^N \left(\sum_{j:(i,j) \in \tilde{E}} x_{i,j}^t - \sum_{j:(j,i) \in \tilde{E}} x_{j,i}^{t-1} \right) + \sum_{j:(j,i) \in \tilde{E}} x_{j,i}^N \right) \\
&= P_1(\tilde{\beta}^{-1}(i)) + \lim_{N \rightarrow \infty} \sum_{j:(j,i) \in \tilde{E}} x_{j,i}^N.
\end{aligned}$$

In Equation 3.13, it was used that

$$\sum_{j:(i,j) \in \tilde{E}} x_{i,j}^1 \stackrel{\text{Flow cons.3.9}}{=} \sum_{j:(j,i) \in \tilde{E}_N^0} x_{j,i}^0 = P_1(\tilde{\beta}^{-1}(i))$$

and that

$$\begin{aligned}
\sum_{j:(i,j) \in \tilde{E}} x_{i,j}^2 &= \sum_{j:(j,i) \in \tilde{E}} x_{j,i}^1 \\
&= \sum_{j:(j,i) \in \tilde{E}} \alpha(j,i) \cdot \sum_{k:(k,j) \in \tilde{E}_N^0} x_{k,j}^0 \\
&= \sum_{j:(j,i) \in \tilde{E}} \alpha(j,i) \cdot P_1(\tilde{\beta}^{-1}(j)) \\
&\stackrel{j \in \tilde{N}_A}{=} 0.
\end{aligned}$$

From the last equation, it can be seen that $x_{i,j}$ satisfies the flow constraint if and only if

$$\lim_{t \rightarrow \infty} \sum_{j:(j,i) \in \tilde{E}} x_{j,i}^t = 0, \quad \forall i \in N_S \setminus \{N_S^W \cup N_S^L\}.$$

Fortunately, this property is always satisfied by SSO-MDPs since

$$\lim_{t \rightarrow \infty} \sum_{j:(j,i) \in \tilde{E}} x_{j,i}^t = \lim_{t \rightarrow \infty} \sum_{j:(j,i) \in \tilde{E}} \mathbb{P}^{d^\infty} \{Z_{t+1} = j, Z_t = i\}$$

where d is derived from the feasible solution of problem 3.9 as described in Theorem 3.6.16. Every policy in an SSO-MDP is proper, which means that the stochastic process reaches with a probability of one in an absorbing state and the probability for a transition at time point t from i to j converges to zero for $t \rightarrow \infty$. Therefore, it is known that

$$\lim_{t \rightarrow \infty} \sum_{j:(j,i) \in \tilde{E}} \mathbb{P}^{d^\infty} \{Z_{t+1} = j, Z_t = i\} = 0, \forall d \in D^{MR}.$$

Observe, that the edge set \tilde{E} of formulation 3.9 does not contain the artificial edges at the absorbing nodes. So, x_{ij} derived from x_{ij}^t , satisfies the flow constraints of the static maximum flow problem 3.12 for $i \in N_S \setminus \{N_S^W \cup N_S^L\}$. It has to be shown that the flow constraint is also satisfied by the defined v_i and x_{ij} at nodes $i \in N_S^W \cup N_S^L$.

$$\begin{aligned} v_i &\stackrel{\text{Def.}}{=} \sum_{t=2}^{\infty} v_i^t \\ &\stackrel{\text{Flow cons. of 3,9}}{=} \sum_{t=2}^{\infty} \sum_{j:(j,i) \in \tilde{E}_N^{t-1}} x_{j,i}^{t-1} \\ &= \sum_{j:(j,i) \in \tilde{E}} \sum_{t=1}^{\infty} x_{j,i}^t \\ &= \sum_{j:(j,i) \in \tilde{E}} \sum_{t=2}^{\infty} x_{j,i}^t + \sum_{j:(j,i) \in \tilde{E}} x_{j,i}^1 \\ &\stackrel{\text{Flow distr. in 3,9}}{=} \sum_{j:(j,i) \in \tilde{E}} \sum_{t=2}^{\infty} x_{j,i}^t + \sum_{j:(j,i) \in \tilde{E}} \alpha(j, i) \cdot \sum_{k:(k,j) \in \tilde{E}_N^0} x_{k,j}^0 \\ &\stackrel{\text{Flow cons. of 3,9}}{=} \sum_{j:(j,i) \in \tilde{E}} \sum_{t=2}^{\infty} x_{j,i}^t + \sum_{j:(j,i) \in \tilde{E}} \alpha(j, i) \cdot \underbrace{P_1(\tilde{\beta}^{-1}(j))}_{=0} \\ &\stackrel{j \in \tilde{N}_A}{=} \sum_{j:(j,i) \in \tilde{E}} \sum_{t=2}^{\infty} x_{j,i}^t \\ &= \sum_{j:(j,i) \in \tilde{E}} \sum_{t \in \mathbb{N}} x_{j,i}^{t+1} \\ &= \sum_{j:(j,i) \in \tilde{E}} \alpha(j, i) \sum_{k:(k,j) \in \tilde{E}} \sum_{t \in \mathbb{N}} x_{k,j}^t \\ &= \sum_{j:(j,i) \in \tilde{E}} \alpha(j, i) \sum_{k:(k,j) \in \tilde{E}} x_{k,j}. \end{aligned}$$

Finally, it is shown that for those pairs of feasible solutions the objective function values differ

exactly by the constant $\sum_{i \in N_S^W} P_1(\tilde{\beta}^{-1}(i))$:

$$\begin{aligned} \sum_{t \in \mathbb{N}} \sum_{i \in N_S^W} v_i^t &= \sum_{t=2}^{\infty} \sum_{i \in N_S^W} v_i^t + \sum_{i \in N_S^W} v_i^1 \\ &= \sum_{i \in N_S^W} v_i + \sum_{i \in N_S^W} \sum_{j: (j,i) \in \bar{E}_{\mathbb{N}}^0} x_{j,i}^0 \\ &= \sum_{i \in N_S^W} v_i + \sum_{i \in N_S^W} P_1(\tilde{\beta}^{-1}(i)). \quad \blacksquare \end{aligned}$$

The subsequent Example 3.6 shows that if there are non-proper policies Proposition 3.6.19 does not hold.

Example 3.6:

Figure 3.7 shows a modified version of Example 3.2 which is no SSO-MDP any more. Assume that $P_1(s_1) = 1$. Then, a feasible solution of the time-expanded maximum flow problem is:

$$\begin{aligned} x_{s,1}^0 &= 1, x_{s,2}^0 = 0, x_{s,3}^0 = 0, x_{s,4}^0 = 0, \\ x_{1,2}^{2t+1} &= 1, x_{2,1}^{2t} = 1, \quad \forall t \in \mathbb{N}_0, \\ \delta(2) &= 1, \\ v_3^t &= v_4^t = 0, \quad \forall t \in \mathbb{N}. \end{aligned}$$

However, the static maximum flow problem is

$$\left| \begin{array}{rcl} \max v_4 & & \\ x_{1,2} - 1 \cdot x_{1,2} & = & 1, \\ 0 & = & -v_3, \\ 0 & = & -v_4, \\ x_{1,2} & \geq & 0. \end{array} \right| \quad (3.14)$$

It has no solution since $x_{1,2} - 1 \cdot x_{1,2} = 0 \neq 1$. *

The goal of this subsection is to determine whether the static maximum flow problem can be used to find an optimal decision rule for a given SSO-MDP. So, it is interesting to investigate whether a time-dependent solution feasible for the time-dependent maximum flow problem can be derived from a feasible solution of the static maximum flow problem with the same objective value – except for a constant. This is more or less the inverse statement of the previous theorem.

From the static solution, a time-dependent solution can be calculated by starting with the given start distribution and distributing the flow according to the distribution of the static solution. Since every policy in an SSO-MDP is proper, every node is connected to an absorbing node. Therefore, only a fraction smaller 1 of a flow entering a circle may stay in that circle. This property is important such that the sum of all $x_{i,j}^t$, $t \in \mathbb{N}$ converges to the static solution $x_{i,j}$ and the objective values of both solutions differ only by $\sum_{i \in N_S^W} P_1(\tilde{\beta}^{-1}(i))$.

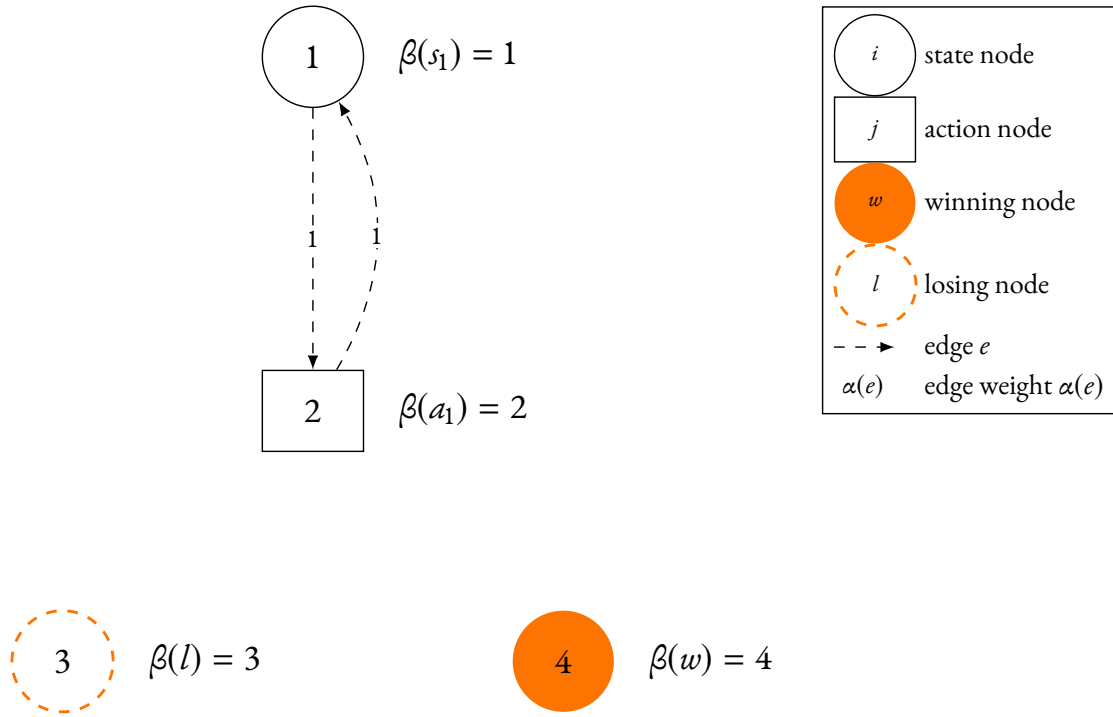


Figure 3.7: Flow network example of an SSO-MDP

Proposition 3.6.20:

Assume the flow network $(\tilde{N}, \tilde{E}, \tilde{\beta}, \alpha)$ of an associated graph of an SSO-MDP and an initial distribution P_1 of the starting state. Let

$$\{x_{i,j}, (i,j) \in \tilde{E}, i \in N_S\}, \quad \text{be feasible for 3.12,}$$

$$\{v_i, i \in N_S^W \cup N_S^L\}$$

where 3.12 is defined for $(\tilde{N}, \tilde{E}, \tilde{\beta}, \alpha)$. Define

$$\delta(j) := \begin{cases} \frac{x_{i,j}}{\sum_{k:(i,k) \in \tilde{E}} x_{i,k}} & \text{for all } (i,j) \in \tilde{E}, i \in N_S \text{ with} \\ \text{arbitrary s.t.} & \sum_{k:(i,k) \in \tilde{E}} x_{i,k} > 0, \\ \sum_{j:(i,j) \in \tilde{E}} \delta(j) = 1, \delta(j) \geq 0 & \text{else.} \end{cases}$$

$$\sum_{j:(j,i) \in \tilde{E}_N^0} x_{j,i}^0 := P_1(\tilde{\beta}^{-1}(i)), \quad \forall i \in N_S,$$

$$x_{i,j}^t := \begin{cases} \alpha(i,j) \cdot \sum_{k:(k,i) \in \tilde{E}_N^{t-1}} x_{k,i}^{t-1}, & \forall (i,j) \in \tilde{E}, i \in \tilde{N}_A, t \in \mathbb{N}, \\ \delta(j) \cdot \sum_{k:(k,i) \in \tilde{E}_N^{t-1}} x_{k,i}^{t-1}, & \forall (i,j) \in \tilde{E}, i \in N_S, t \in \mathbb{N}. \end{cases}$$

$$v_i^t := \sum_{j:(j,i) \in \tilde{E}_N^{t-1}} x_{j,i}^{t-1}, \quad \forall i \in N_S^W \cup N_S^L, t \in \mathbb{N}.$$

Then, $x_{i,j}^t$, $\delta(j)$ and v_i^t are feasible for 3.9 defined for the time-expanded flow network of $(\tilde{N}, \tilde{E}, \tilde{\beta}, \alpha)$ and

$$\sum_{t \in \mathbb{N}} \sum_{i \in N_S^W} v_i^t = \sum_{i \in N_S^W} v_i + \sum_{i \in N_S^W} P_1(\tilde{\beta}^{-1}(i)).$$

PROOF. Let $x_{i,j}$ and v^t be feasible for 3.12 and define $x_{i,j}^t$, $\delta(j)$ and v_i^t as described in the proposition.

First, it is shown that $\delta(j)$ specifies a probability distribution over all j with $(i, j) \in \tilde{E} = \tilde{E}_{\tilde{N}}^t$, $t \in \mathbb{N}$ for a fixed $i \in N_S$. Consider a state node $i \in N_S$ with $\sum_{k:(k,i) \in \tilde{E}} x_{k,i} > 0$. As $x_{i,j} \geq 0$ holds, the non-negativity of $\delta(j)$ is given. Furthermore,

$$\begin{aligned} \sum_{j:(i,j) \in \tilde{E}} \delta(j) &= \sum_{j:(i,j) \in \tilde{E}} \frac{x_{i,j}}{\sum_{k:(i,k) \in \tilde{E}} x_{i,k}} \\ &= \frac{\sum_{j:(i,j) \in \tilde{E}} x_{i,j}}{\sum_{k:(i,k) \in \tilde{E}} x_{i,k}} = 1 \end{aligned}$$

holds. In all other state nodes $i \in N_S$, δ is set to an arbitrary probability distribution over the outgoing edges $(i, j) \in \tilde{E}$.

It needs to be checked that $x_{i,j}^t$ satisfies the flow conditions of 3.9. The flow distribution constraints of 3.9 are obviously satisfied by the definition of $x_{i,j}^t$ in the proposition. For $i \in \tilde{N}_A$, $t \in \mathbb{N}$, it holds:

$$\begin{aligned} \sum_{j:(i,j) \in \tilde{E}} x_{i,j}^t &\stackrel{\text{Def. } x_{i,j}^t}{=} \sum_{j:(i,j) \in \tilde{E}} \left(\alpha(i, j) \cdot \sum_{k:(k,i) \in \tilde{E}_{\tilde{N}}^{t-1}} x_{k,i}^{t-1} \right) \\ &= \left(\sum_{j:(i,j) \in \tilde{E}} \alpha(i, j) \right) \cdot \left(\sum_{k:(k,i) \in \tilde{E}_{\tilde{N}}^{t-1}} x_{k,i}^{t-1} \right) \\ &= \sum_{k:(k,i) \in \tilde{E}_{\tilde{N}}^{t-1}} x_{k,i}^{t-1}. \end{aligned}$$

And for $i \in N_S$ analogous transformations are possible as it was shown that $\delta(j)$ specifies a probability distribution over all j with $(i, j) \in \tilde{E}$ for a fixed $i \in N_S$.

It remains to show that the objective values of those pairs of corresponding solutions are identical. If it can be shown that

$$\sum_{t \in \mathbb{N}} \sum_{j:(i,j) \in \tilde{E}} x_{i,j}^t = \sum_{j:(i,j) \in \tilde{E}} x_{i,j}, \quad i \in N_S, \quad (3.15)$$

this can be used to show that the objective functions are identical up to the constant $\sum_{i \in N_S^W} P_1(\tilde{\beta}^{-1}(i))$:

$$\begin{aligned} &\sum_{t \in \mathbb{N}} \sum_{i \in N_S^W} v_i^t - \sum_{i \in N_S^W} P_1(\tilde{\beta}^{-1}(i)) \\ &\stackrel{\text{Def. } v_i^t}{=} \sum_{t \in \mathbb{N}} \sum_{i \in N_S^W} \sum_{j:(j,i) \in \tilde{E}_{\tilde{N}}^{t-1}} x_{j,i}^{t-1} - \sum_{i \in N_S^W} \sum_{j:(j,i) \in \tilde{E}_{\tilde{N}}^0} x_{j,i}^0 \end{aligned} \quad (3.16)$$

$$\begin{aligned}
&= \sum_{i \in N_S^W} \sum_{j: (j,i) \in \tilde{E}} \sum_{t=1}^{\infty} x_{j,i}^t \\
\stackrel{\text{Def. } x^t}{=} & \sum_{i \in N_S^W} \sum_{j: (j,i) \in \tilde{E}} \sum_{t=1}^{\infty} \alpha(j, i) \sum_{k: (k,j) \in \tilde{E}_{\tilde{N}}^{t-1}} x_{k,j}^{t-1} \\
&= \sum_{i \in N_S^W} \sum_{j: (j,i) \in \tilde{E}} \alpha(j, i) \cdot \left(\sum_{t=1}^{\infty} \sum_{k: (k,j) \in \tilde{E}_{\tilde{N}}^t} x_{k,j}^t + \underbrace{\sum_{k: (k,j) \in \tilde{E}_{\tilde{N}}^0} x_{k,j}^0}_{=0} \right) \tag{3.17} \\
\stackrel{\text{Def. } x_{k,j}^t}{=} & \sum_{i \in N_S^W} \sum_{j: (j,i) \in \tilde{E}} \alpha(j, i) \cdot \left(\sum_{t=1}^{\infty} \sum_{k: (k,j) \in \tilde{E}} \left(\delta(j) \cdot \sum_{l: (l,k) \in \tilde{E}_{\tilde{N}}^{t-1}} x_{l,k}^{t-1} \right) \right) \\
\stackrel{\text{Flow cons. 3.9}}{=} & \sum_{i \in N_S^W} \sum_{j: (j,i) \in \tilde{E}} \alpha(j, i) \cdot \left(\sum_{t=1}^{\infty} \sum_{k: (k,j) \in \tilde{E}} \left(\delta(j) \cdot \sum_{l: (k,l) \in \tilde{E}} x_{k,l}^t \right) \right) \\
&= \sum_{i \in N_S^W} \sum_{j: (j,i) \in \tilde{E}} \alpha(j, i) \cdot \sum_{k: (k,j) \in \tilde{E}} \left(\delta(j) \cdot \sum_{t=1}^{\infty} \sum_{l: (k,l) \in \tilde{E}} x_{k,l}^t \right) \tag{3.18} \\
\stackrel{\text{Equ. 3.15}}{=} & \sum_{i \in N_S^W} \sum_{j: (j,i) \in \tilde{E}} \alpha(j, i) \cdot \sum_{k: (k,j) \in \tilde{E}} \left(\delta(j) \cdot \sum_{l: (k,l) \in \tilde{E}} x_{k,l} \right) \\
\stackrel{\text{Def. } \delta(j)}{=} & \sum_{i \in N_S^W} \sum_{j: (j,i) \in \tilde{E}} \alpha(j, i) \cdot \sum_{k: (k,j) \in \tilde{E}, \text{ with } \sum_{l: (k,l) \in \tilde{E}} x_{k,l} > 0} \left(\frac{x_{k,j}}{\sum_{l: (k,l) \in \tilde{E}} x_{k,l}} \cdot \sum_{l: (k,l) \in \tilde{E}} x_{k,l} \right) \\
&= \sum_{i \in N_S^W} \sum_{j: (j,i) \in \tilde{E}} \alpha(j, i) \cdot \sum_{k: (k,j) \in \tilde{E}} x_{k,j} \\
\stackrel{x \text{ feasible}}{=} & \sum_{i \in N_S^W} v_i.
\end{aligned}$$

In Equation 3.17, it was used that j must be an action node, if i is a state node and (i, j) is an edge in \tilde{E} . As the distribution of the starting state is a distribution only over states, it holds $P_1(\tilde{\beta}^{-1}(j)) = 0, \forall j \in \tilde{N}_{\mathcal{A}}$. In Equation 3.18, k is a state node and $\sum_{t \in \mathbb{N}} \sum_{l: (k,l) \in \tilde{E}} x_{k,l}$ can be substituted by $\sum_{l: (k,l) \in \tilde{E}} x_{k,l}$.

So, it remains to show Equation 3.15. First, assume an acyclic static flow network and prove the statement by induction over a topological ordering of the state nodes. As a forward reference to the next Section, there always exists an equivalent acyclic representation of an SSO-MDP. Afterwards, Equation 3.15 is proven for static flow networks that contain circles.

Assume that the associated flow network $(\tilde{N}, \tilde{E}, \tilde{\beta}, \alpha)$ of the SSO-MDP is acyclic. Let (i_1, \dots, i_n)

be a topological ordering of the state nodes such that for all paths $i-j-i' = ((i, j), (j, i'))$ that connect two state nodes $i, i' \in N_S$ the node i is a predecessor of i' in the topological ordering. Observe that in paths of an associated graph of an SSO-MDP, there always exists an action node between two state nodes. In the example path j is an action node.

Equation 3.15 is proven by induction over the topological ordering (i_1, \dots, i_n) .

Induction start: For i_1 , the set of predecessors $\{i \mid \exists j \in \tilde{N}_A, (i, j) \in \tilde{E}, t \in \mathbb{N}, (j, i) \in \tilde{E}\}$ is empty. Together with the feasibility of $x_{i,j}$ for the static flow constraint, it holds

$$\sum_{j:(i_1,j) \in \tilde{E}} x_{i_1,j} \stackrel{\text{feas.}}{=} P_1(\tilde{\beta}^{-1}(i_1)) + \sum_{j:(j,i_1) \in \tilde{E}} \alpha(j, i_1) \underbrace{\sum_{k:(k,j) \in \tilde{E}} x_{k,j}}_{=0} = P_1(\tilde{\beta}^{-1}(i_1)).$$

The feasibility of $x_{i,j}^t$ for the flow constraints of the time-expanded maximum flow formulation is used and that $\sum_{k:(k,j) \in \tilde{E}_{\tilde{N}}^0} x_{k,j}^0 = P_1(\tilde{\beta}^{-1}(j)) = 0$ is satisfied for action nodes j . This leads to

$$\begin{aligned} \sum_{t \in \mathbb{N}} \sum_{j:(i_1,j) \in \tilde{E}} x_{i_1,j}^t &\stackrel{\text{Flow cons. 3.9}}{=} \sum_{t \in \mathbb{N}} \sum_{j:(j,i_1) \in \tilde{E}_{\tilde{N}}^{t-1}} x_{j,i_1}^{t-1} \\ &= \sum_{t \in \mathbb{N}} \sum_{j:(j,i_1) \in \tilde{E}} x_{j,i_1}^t + \sum_{j:(j,i_1) \in \tilde{E}_{\tilde{N}}^0} x_{j,i_1}^0 \\ &\stackrel{\text{Def. } x_{i,j}^t}{=} \sum_{j:(j,i_1) \in \tilde{E}} \alpha(j, i_1) \left(\underbrace{\sum_{t \in \mathbb{N}} \sum_{k:(k,j) \in \tilde{E}} x_{k,j}^t}_{=0} + \underbrace{\sum_{k:(k,j) \in \tilde{E}_{\tilde{N}}^0} x_{k,j}^0}_{=0} \right) + P_1(\tilde{\beta}^{-1}(i_1)) \\ &\stackrel{i_1 \text{ no Pred.}}{=} P_1(\tilde{\beta}^{-1}(i_1)). \end{aligned}$$

So, the induction hypothesis holds for i_1 .

Induction hypothesis: Assume for all predecessors of i_k Equation 3.15 holds.

Induction step: Then,

$$\begin{aligned} &\sum_{t \in \mathbb{N}} \sum_{j:(i_k,j) \in \tilde{E}} x_{i_k,j}^t \\ &\stackrel{\text{Flow cons. 3.9}}{=} \sum_{t \in \mathbb{N}} \sum_{j:(j,i_k) \in \tilde{E}_{\tilde{N}}^{t-1}} x_{j,i_k}^{t-1} \\ &= \sum_{t \in \mathbb{N}} \sum_{j:(j,i_k) \in \tilde{E}} x_{j,i_k}^t + \sum_{j:(j,i_k) \in \tilde{E}_{\tilde{N}}^0} x_{j,i_k}^0 \end{aligned}$$

$$\begin{aligned}
& \stackrel{\text{Def. } x_{ij}^t}{=} \sum_{j:(j,i_k) \in \tilde{E}} \alpha(j, i_k) \left(\sum_{t \in \mathbb{N}} \sum_{k:(k,j) \in \tilde{E}} x_{k,j}^t + \underbrace{\sum_{k:(k,j) \in \tilde{E}_{\tilde{N}}^0} x_{k,j}^0}_{=0} \right) + P_1(\tilde{\beta}^{-1}(i_k)) \quad (3.19) \\
& \stackrel{(\text{IH})}{=} \sum_{j:(j,i_k) \in \tilde{E}} \alpha(j, i_k) \sum_{k:(k,j) \in \tilde{E}} x_{k,j} + P_1(\tilde{\beta}^{-1}(i_k)).
\end{aligned}$$

In Equation 3.19, it was used again that j is an action node and $P_1(\tilde{\beta}^{-1}(j)) = 0$, $\forall j \in \tilde{N}_A$. Also, the same transformation need to be applied as in Equation 3.17 to apply Equation 3.15. The induction hypothesis can be applied since k is a state node and a predecessor of i_k .

Equation 3.15 holds also for flow networks that contain circles. Let

$$C = (i_1, j_2, i_3, \dots, j_k, i_1)$$

be an elementary circle in the flow network of an SSO-MDP. For a better understanding of the idea and not getting lost in the notation, some assumptions on the circle are made. At the end of the proof, it will be argued why this proof applies also to generalized circles. Assume that only node i_1 has an incoming edge from a node not contained in the circle and that i_1 is the only state node of the circle for which $P(\tilde{\beta}^{-1}(i_1)) > 0$ holds. Furthermore, let p_C be the circle path $i_1 - j_2 - \dots - j_k - i_1$ and $\alpha(p_C)$ the product $\delta(j_2) \cdot \alpha(j_2, i_3) \cdot \dots \cdot \delta(j_k) \cdot \alpha(j_k, i_1)$. And assume for a simpler notation that $i \in C$ is only connected to i_1 via the circle path of C .

Due to the assumption $P(\tilde{\beta}^{-1}(i_1)) > 0$, for each state node i of the circle $\sum_{k:(i,k) \in \tilde{E}} x_{i,k} > 0$ holds and $\delta(j)$ is therefore for all state nodes of the circle defined as the fraction of flow that goes along the edge (i, j) , which will be use at some point in the proof.

Under the made assumptions, the following holds:

$$\begin{aligned}
& \sum_{t \in \mathbb{N}} \sum_{j:(i_1,j) \in \tilde{E}} x_{i_1,j}^t \\
\text{Flow cons. 3.9} & \stackrel{=}{=} \sum_{t \in \mathbb{N}} \sum_{j:(j,i_1) \in \tilde{E}_{\tilde{N}}^{t-1}} x_{j,i_1}^{t-1} \\
& = \sum_{t \in \mathbb{N}} \sum_{j:(j,i_1) \in \tilde{E}} x_{j,i_1}^t + \sum_{j:(j,i_1) \in \tilde{E}_{\tilde{N}}^0} x_{j,i_1}^0 \\
& \stackrel{\text{Def. } x_{ij}^t}{=} \sum_{j:(j,i_1) \in \tilde{E}} \alpha(j, i_1) \left(\sum_{t \in \mathbb{N}} \sum_{k:(k,j) \in \tilde{E}} x_{k,j}^t + \underbrace{\sum_{k:(k,j) \in \tilde{E}_{\tilde{N}}^0} x_{k,j}^0}_{=0} \right) + P_1(\tilde{\beta}^{-1}(i_1))
\end{aligned}$$

$$\begin{aligned}
& \stackrel{j \in \tilde{N}_A}{=} \sum_{j:(j,i_1) \in \tilde{E}} \alpha(j, i_1) \left(\sum_{t \in \mathbb{N}} \sum_{k:(k,j) \in \tilde{E}} x_{k,j}^t \right) + P_1(\tilde{\beta}^{-1}(i_1)) \\
& = \sum_{j:(j,i_1) \in \tilde{E}} \alpha(j, i_1) \left(\sum_{t \in \mathbb{N}} \sum_{k:(k,j) \in \tilde{E}, k \in C} x_{k,j}^t + \sum_{t \in \mathbb{N}} \sum_{k:(k,j) \in \tilde{E}, k \notin C} x_{k,j}^t \right) + P_1(\tilde{\beta}^{-1}(i_1)) \\
& \stackrel{\text{Ass. } C}{=} \alpha(j_k, i_1) \sum_{t \in \mathbb{N}} x_{i_{k-1}, j_k}^t + \sum_{j:(j,i_1) \in \tilde{E}} \alpha(j, i_1) \left(\sum_{t \in \mathbb{N}} \sum_{k:(k,j) \in \tilde{E}, k \notin C} x_{k,j}^t \right) + P_1(\tilde{\beta}^{-1}(i_1)) \\
& \stackrel{\text{Def. } x_{ij}^t}{=} \alpha(j_k, i_1) \cdot \sum_{t \in \mathbb{N}} x_{i_{k-1}, j_k}^t + \sum_{j:(j,i_1) \in \tilde{E}} \alpha(j, i_1) \left(\sum_{t \in \mathbb{N}} \sum_{k:(k,j) \in \tilde{E}, k \notin C} \delta(j) \sum_{r:(r,k) \in \tilde{E}_{\tilde{N}}^{t-1}} x_{r,k}^{t-1} \right) + P_1(\tilde{\beta}^{-1}(i_1)) \\
& \stackrel{\text{Flow cons. 3.9}}{=} \alpha(j_k, i_1) \cdot \sum_{t \in \mathbb{N}} x_{i_{k-1}, j_k}^t + \sum_{j:(j,i_1) \in \tilde{E}} \alpha(j, i_1) \left(\sum_{k:(k,j) \in \tilde{E}, k \notin C} \delta(j) \sum_{t \in \mathbb{N}} \sum_{r:(k,r) \in \tilde{E}} x_{k,r}^t \right) + P_1(\tilde{\beta}^{-1}(i_1)) \\
& \stackrel{(\text{IH})}{=} \alpha(j_k, i_1) \cdot \sum_{t \in \mathbb{N}} x_{i_{k-1}, j_k}^t + \underbrace{\sum_{j:(j,i_1) \in \tilde{E}} \alpha(j, i_1) \left(\sum_{k:(k,j) \in \tilde{E}, k \notin C} \delta(j) \sum_{r:(k,r) \in \tilde{E}} x_{k,r} \right)}_{=: K} + P_1(\tilde{\beta}^{-1}(i_1)) \\
& \stackrel{\text{Def. } x_{ij}^t}{=} \alpha(j_k, i_1) \cdot \sum_{t \in \mathbb{N}} \delta(j_k) \sum_{j:(j,i_{k-1}) \in \tilde{E}_{\tilde{N}}^{t-1}} x_{j,i_{k-1}}^{t-1} + K \\
& \stackrel{\text{Ass. } C}{=} \alpha(j_k, i_1) \cdot \delta(j_k) \cdot \sum_{t \in \mathbb{N}} x_{j_{k-2}, i_{k-1}}^t + K \tag{3.20} \\
& = \alpha(j_k, i_1) \cdot \delta(j_k) \cdot \dots \cdot \alpha(j_2, i_3) \cdot \sum_{t \in \mathbb{N}} x_{i_1, j_2}^t + K \\
& = \alpha(j_k, i_1) \cdot \delta(j_k) \cdot \dots \cdot \alpha(j_2, i_3) \delta(j_2) \cdot \sum_{t \in \mathbb{N}} \sum_{j:(j,i_1) \in \tilde{E}_{\tilde{N}}^{t-1}} x_{j,i_1}^{t-1} + K \\
& \stackrel{\text{Flow cons. 3.9}}{=} \alpha(p_C) \cdot \sum_{t \in \mathbb{N}} \sum_{j:(i_1, j) \in \tilde{E}} x_{i_1, j}^t + K
\end{aligned}$$

In Equation 3.20, the assumption that all state nodes $i \in C$, $i \neq i_1$ do not have any incoming edge from outside the circle and the assumption that there exists a unique path between the nodes of the circle were used.

Altogether, a recursive equation for $\sum_{t \in \mathbb{N}} \sum_{j:(i_1, j) \in \tilde{E}} x_{i_1, j}^t$ was found. Observe that $\alpha(p_C) < 1$ holds since we face an SSO-MDP. So, the geometric sum can be applied to evaluate an infinite application of this recursive equation:

$$\sum_{t \in \mathbb{N}} \sum_{j:(i_1, j) \in \tilde{E}} x_{i_1, j}^t$$

$$\begin{aligned}
&= K + \alpha(p_C) \cdot \sum_{t \in \mathbb{N}} \sum_{j:(i_1,j) \in \tilde{E}} x_{i_1,j}^t \\
&= K + \alpha(p_C) \cdot \left(K + \alpha(p_C) \cdot \sum_{t \in \mathbb{N}} \sum_{j:(i_1,j) \in \tilde{E}} x_{i_1,j}^t \right) \\
&= K \cdot \sum_{k=0}^{\infty} \alpha(p_C)^k \\
&\stackrel{\text{Geom. series}}{=} K \cdot \frac{1}{1 - \alpha(p_C)}
\end{aligned}$$

For the right side $\sum_{j:(i_1,j) \in \tilde{E}} x_{i_1,j}$ of Equation 3.15, which is currently proved, it holds

$$\begin{aligned}
&\sum_{j:(i_1,j) \in \tilde{E}} x_{i_1,j} \\
&\stackrel{\text{Flow cons. 3.12}}{=} P_1(\tilde{\beta}^{-1}(i_1)) + \sum_{j:(j,i_1) \in \tilde{E}} \alpha(j, i_1) \sum_{k:(k,j) \in \tilde{E}} x_{k,j} \\
&= \underbrace{P_1(\tilde{\beta}^{-1}(i_1)) + \sum_{j:(j,i_1) \in \tilde{E}} \alpha(j, i_1) \sum_{k:(k,j) \in \tilde{E}, k \notin C} x_{k,j} + \sum_{j:(j,i_1) \in \tilde{E}} \alpha(j, i_1) \sum_{k:(k,j) \in \tilde{E}, k \in C} x_{k,j}}_{=:K'} \\
&= K' + \sum_{j:(j,i_1) \in \tilde{E}} \alpha(j, i_1) \sum_{k:(k,j) \in \tilde{E}, k \in C} x_{k,j} \\
&\stackrel{\text{Ass. C}}{=} K' + \alpha(j_k, i_1) \cdot x_{i_{k-1},j_k} \\
&\stackrel{\text{mult. by 1}}{=} K' + \alpha(j_k, i_1) \cdot \frac{x_{i_{k-1},j_k}}{\sum_{j:(i_{k-1},j) \in \tilde{E}} x_{i_{k-1},j}} \cdot \sum_{j:(i_{k-1},j) \in \tilde{E}} x_{i_{k-1},j} \\
&\stackrel{\text{Def. } \delta(j)}{=} K' + \alpha(j_k, i_1) \cdot \delta(j_k) \cdot \sum_{j:(i_{k-1},j) \in \tilde{E}} x_{i_{k-1},j} \\
&= K' + \alpha(j_k, i_1) \cdot \delta(j_k) \cdot \dots \cdot \delta(j_2) \sum_{j:(i_1,j) \in \tilde{E}} x_{i_1,j} \\
&= K' + \alpha(p_C) \cdot \sum_{j:(i_1,j) \in \tilde{E}} x_{i_1,j}
\end{aligned}$$

So, an analogous recursive equation for $\sum_{j:(i_1,j) \in \tilde{E}} x_{i_1,j}$ is found. Again, the limit of the geometric series can be used to evaluate an infinite application of this equation:

$$\begin{aligned}
&\sum_{j:(i_1,j) \in \tilde{E}} x_{i_1,j} \\
&= K' + \alpha(p_C) \cdot \sum_{j:(i_1,j) \in \tilde{E}} x_{i_1,j}
\end{aligned}$$

$$\begin{aligned}
&= K' + \alpha(p_C) \cdot \left(K' + \alpha(p_C) \cdot \sum_{j:(i_1,j) \in \tilde{E}} x_{i_1,j} \right) \\
&= K' \cdot \sum_{k=0}^{\infty} \alpha(p_C)^k \\
&\stackrel{\text{Geom. series}}{=} K' \cdot \frac{1}{1 - \alpha(p_C)}
\end{aligned}$$

Finally, it remains to show $K = K'$:

$$\begin{aligned}
K &= \sum_{j:(j,i_1) \in \tilde{E}} \alpha(j, i_1) \left(\sum_{k:(k,j) \in \tilde{E}, k \notin C} \delta(j) \sum_{r:(k,r) \in \tilde{E}} x_{k,r} \right) + P_1(\tilde{\beta}^{-1}(i_1)) \\
&\stackrel{\text{Def. } \delta(j)}{=} \sum_{j:(j,i_1) \in \tilde{E}} \alpha(j, i_1) \left(\sum_{k:(k,j) \in \tilde{E}, k \notin C} \frac{x_{k,j}}{\sum_{r:(k,r) \in \tilde{E}} x_{k,r}} \sum_{r:(k,r) \in \tilde{E}} x_{k,r} \right) + P_1(\tilde{\beta}^{-1}(i_1)) \\
&= \sum_{j:(j,i_1) \in \tilde{E}} \alpha(j, i_1) \sum_{k:(k,j) \in \tilde{E}, k \notin C} x_{k,j} + P_1(\tilde{\beta}^{-1}(i_1)) \\
&= K'.
\end{aligned}$$

The idea of the proof is that there exists some incoming flow from outside the circle for which the induction hypothesis can be applied and for the flow staying inside the circle the limit of the geometric series can be applied. The proof was done for a simplified circle. However, it is possible to generalize this result to an arbitrary circle. If there is another node $i \in C$, $i \neq i_1$ that has an incoming path from a state node not in the circle, the induction hypothesis can be applied to its predecessor and the incoming flow to the circle can be encapsulated in the constant K respectively K' . If the second assumption is relaxed and there is a path from a circle node $i \in C$ to another circle node $j \in C$, the flow incoming to node i can still be expressed by a factor times the flow outgoing of i_1 . Since C contains only finitely many nodes, we will at some step return in i_1 and therefore the incoming flow of i can be expressed by a constant that is less than 1 times the outgoing flow of i_1 . So, the geometric series can still be applied. ■

Theorem 3.6.21 (Feasible solution of static maximum flow problem characterizes value of policy):
Assume the flow network $(\tilde{N}, \tilde{E}, \tilde{\beta}, \alpha)$ of an associated graph of an SSO-MDP and an initial distribution P_1 of the starting state. Let $x_{i,j}$ and v_i be a feasible solution of the linear program 3.12 defined for that network. Define a decision rule $d \in D^{MR}$ as

$$\mathbb{P}\{d(s) = a\} := \begin{cases} \frac{x_{i,j}}{\sum_{k:(k,i) \in \tilde{E}} x_{k,i}} & \text{for all } (i,j) \in \tilde{E}, i \in N_S \text{ s.t.} \\ & \sum_{k:(k,i) \in \tilde{E}} x_{k,i} > 0, \\ q_s(a) \text{ arbitrary s.t.} & \text{else.} \\ \sum_{a \in \mathcal{A}} q_s(a) = 1, & \\ q_s(a) \geq 0, & \end{cases}$$

for all $a \in \mathcal{A}$, where $j = \tilde{\beta}(a)$ and $i = \tilde{\beta}(s)$. Then,

$$\sum_{s \in S \setminus W} P_1(s) v^{d^\infty}(s) = \sum_{i \in N_S^W} v_i.$$

PROOF. Define d as described in the theorem and observe that

$$\mathbb{P}\{d(\tilde{\beta}^{-1}(i)) = \tilde{\beta}^{-1}(j)\} = \delta(j), \quad \forall (i, j) \in \tilde{E}, \quad i \in N_S \text{ with } \sum_{k: (i,k) \in \tilde{E}} x_{i,k} > 0,$$

for $\delta(j)$ defined according to Proposition 3.6.20. Following Proposition 3.6.20, from $x_{i,j}$ and v_i a feasible solution $x_{i,j}^t$ and v_i^t can be constructed for program 3.9. For the objective value of that feasible solution holds

$$\sum_{i \in N_S^W} v_i + \sum_{i \in N_S^W} P_1(\tilde{\beta}^{-1}(i)) \stackrel{\text{Prop. 3.6.20}}{=} \sum_{t \in \mathbb{N}} \sum_{i \in N_S^W} v_i^t.$$

From Corollary 3.6.17, it can be derived that the objective value equals the value of the stationary policy d^∞ under the initial distribution P_1 :

$$\sum_{t \in \mathbb{N}} \sum_{i \in N_S^W} v_i^t \stackrel{\text{Cor. 3.6.17}}{=} \sum_{s \in S \setminus W} v^{d^\infty}(s) \cdot P_1(s) + \sum_{s \in W} P_1(s).$$

Since $\sum_{i \in N_S^W} P_1(\tilde{\beta}^{-1}(i)) = \sum_{s \in W} P_1(s)$, the desired result holds. \blacksquare

Example 3.7 (Max flow formulation):

Consider the time-expanded graph of Figure 3.6, which is associated with the SSO-MDP of Example 3.2. The maximum flow formulation for this example is:

$$\left| \begin{array}{l} \max v_4 \\ x_{1,2} - x_{2,1} = 1, \\ x_{2,1} + x_{2,3} + x_{2,4} - x_{1,2} = 0, \\ -x_{2,3} = -v_3, \\ -x_{2,4} = -v_4, \\ x_{2,1} - 0.6 \cdot x_{1,2} = 0, \\ x_{2,3} - 0.2 \cdot x_{1,2} = 0, \\ x_{2,4} - 0.2 \cdot x_{1,2} = 0, \\ x_{1,2} - \delta(2) \cdot (x_{2,1} + 1) = 0, \\ x_{1,2}, x_{2,1}, x_{2,3}, x_{2,4} \geq 0, \end{array} \right|$$

The equivalent refined formulation is:

$$\left| \begin{array}{rcl} \max v_4 & & \\ x_{1,2} - 0.6 \cdot x_{1,2} & = & 1, \\ -0.2 \cdot x_{1,2} & = & -v_3, \\ -0.2 \cdot x_{1,2} & = & -v_4, \\ x_{1,2} & \geq & 0, \end{array} \right|$$

The unique solution of this linear program is $x_{1,2}^* = 2.5$. The corresponding decision rule is

$$\mathbb{P}^{d^\infty} \{d(s_1) = a_1\} = \frac{2.5}{2.5} = 1.$$

The probability of winning the SSO-MDP is $v_4 = 0.5$. *

Finally, all results of this and the last subsection can be combined to get the main result. Theorem 3.6.22 shows the correspondence between a stationary policy of an SSO-MDP and a feasible solution of the static maximum flow problem.

Theorem 3.6.22 (Stationary policy corresponds to solution of static maximum flow problem):
Assume the flow network $(\tilde{N}, \tilde{E}, \tilde{\beta}, \alpha)$ of an associated graph of an SSO-MDP and an initial distribution P_1 of the starting state.

Then, for every stationary policy d^∞ of the SSO-MDP there exists a feasible solution of the linear program 3.12 and vice versa. Furthermore, it holds for the value of the policy and the objective value of the feasible solution that

$$\sum_{s \in S \setminus W} v^{d^\infty}(s) \cdot P_1(s) = \sum_{i \in N_S^W} v_i.$$

PROOF. “ \implies ” Assume a stationary policy d^∞ of the SSO-MDP. Define

$$x_{i,j}^t := \mathbb{P}^{d^\infty} \{Z_{t+1} = j, Z_t = i\}, \quad \forall t \in \mathbb{N}, \quad \forall (i,j) \in \tilde{E} \subseteq E$$

as in Definition 3.6.9. Then, by Theorem 3.6.11, $x_{i,j}^t$ satisfies the flow constraints of the time-expanded maximum flow problem 3.9 for all $t \geq 2$, for all $i \in \tilde{N} \setminus \{N_S^W \cup N_S^L\}$, which is a subset of N . Also, the edge set \tilde{E} differs from E only by the edges adjacent to the artificial action nodes at the absorbing states. Define

$$\tilde{E}_t := \{((t, i), (t+1, j)) \mid (i, j) \in \tilde{E}, t \in \mathbb{N}\}.$$

Add edges $\tilde{E}_0 := \{((0, s), (1, i)), i \in N_S\}$ that connect an artificial source node s with each state node of the time-expanded flow network at time 0 and define

$$x_{s,i}^0 := P_1(\tilde{\beta}^{-1}(i)), \quad \forall i \in \tilde{N}.$$

Thus, $x_{s,i}^0$ satisfies the first constraints of the time-expanded maximum flow problem 3.9. Furthermore, it holds that $\{x_{i,j}^t, t \in \mathbb{N}_0, (i,j) \in \widetilde{E}_{\widetilde{N}}^t\}$ is also valid for the flow constraint at $t = 1$. This can be seen by Proposition 3.6.10 and

$$\sum_{j:(i,j) \in \widetilde{E}_{\widetilde{N}}^1} x_{i,j}^1 \stackrel{\text{Prop. 3.6.10}}{=} \mathbb{P}^{d^{\infty}} \{Z_1 = i\} = P_1(\widetilde{\beta}^{-1}(i)) = x_{s,i}^0 = \sum_{j:(j,i) \in \widetilde{E}_{\widetilde{N}}^0} x_{j,i}^0, \forall i \in \widetilde{N} \setminus \{N_S^W \cup N_S^L\}.$$

Define

$$v_i^t := \sum_{j:(j,i) \in \widetilde{E}_{\widetilde{N}}^{t-1}} x_{j,i}^{t-1}, \forall i \in N_S^W \cup N_S^L, t \in \mathbb{N},$$

such that the flow conditions for the winning and losing nodes of the time-expanded maximum flow problem 3.9 are also satisfied.

It remains to show that the flow distribution constraints are satisfied in the action nodes and in the state nodes such that the constructed solution is feasible for problem 3.9. Assume an action node $i \in N_A$ with $\sum_{j:(j,i) \in E} x_{j,i}^{t-1} > 0$, then by Theorem 3.6.13 it holds that

$$\frac{x_{i,j}^t}{\sum_{k:(k,i) \in E} x_{k,i}^{t-1}} = p(\beta^{-1}(j) | \beta^{-1}(i)) = \alpha(i,j), \forall (i,j) \in \widetilde{E}_{\widetilde{N}}^t, \forall t \geq 2.$$

With the defined edge set $\widetilde{E}_{\widetilde{N}}^0$, the proof of Theorem 3.6.13 can be extended to $t = 1$. In a state node $i \in N_S$ with $\sum_{j:(j,i) \in E} x_{j,i}^{t-1} > 0$, it is known by Theorem 3.6.12 that

$$\frac{x_{i,j}^t}{\sum_{k:(k,i) \in E} x_{k,i}^{t-1}} = q_{d(\beta^{-1}(i))}(\beta^{-1}(j)), \forall (i,j) \in \widetilde{E}_{\widetilde{N}}^t, \forall t \geq 2.$$

The proof of Theorem 3.6.12 can also be extended to $t = 1$ due to our definition of $\widetilde{E}_{\widetilde{N}}^0$. So, $\delta(j)$ can be defined as

$$\delta(j) := q_{d(\beta^{-1}(i))}(\beta^{-1}(j)), \forall (i,j) \in \widetilde{E}, i \in N_S$$

and the flow distribution constraint is satisfied.

If the condition $\sum_{j:(j,i) \in E} x_{j,i}^{t-1} > 0$ does not hold, the flow distribution constraints are still satisfied. In this case, from the flow conditions it is derived that all outgoing edges $x_{i,j}^t$ are zero and thus it holds

$$\sum_{k:(k,i) \in E} x_{k,i}^{t-1} \cdot \delta(j) = 0 \cdot \delta(j) = 0 = x_{i,j}^t, \forall (i,j) \in E, \forall i \in N_S, \forall \delta(j) \in [0, 1], t \in \mathbb{N}$$

and

$$\sum_{k:(k,i) \in E} x_{k,i}^{t-1} \cdot \alpha(i,j) = 0 \cdot \alpha(i,j) = 0 = x_{i,j}^t, \forall (i,j) \in E, \forall i \in N_A, t \in \mathbb{N}.$$

So it was shown that the defined $x_{i,j}^t$, v_i^t and $\delta(j)$ are a feasible solution for the time-expanded maximum flow problem.

From Corollary 3.6.17, it follows that the objective value of the feasible solution equals the value of the stationary policy d^∞ plus the constant $\sum_{i \in W} P_1(s)$:

$$\sum_{t \in \mathbb{N}} \sum_{i \in N_S^W} v_i^t = \sum_{s \in S \setminus W} v^{d^\infty}(s) \cdot P_1(s) + \sum_{s \in W} P_1(s). \quad (3.21)$$

Note, that the decision rule used in Corollary 3.6.17 was derived from the variable δ and therefore equals the considered decision rule d of the SSO-MDP.

According to Proposition 3.6.19, a feasible solution of the static maximum flow problem can be constructed by

$$\left\{ \begin{array}{l} x_{i,j} := \sum_{t \in \mathbb{N}} x_{i,j}^t, \quad (i,j) \in \tilde{E}, \quad i \in N_S \\ v_i := \sum_{t=2}^{\infty} v_i^t, \quad i \in N_S^W \cup N_S^L \end{array} \right\}.$$

For this feasible solution of the static maximum flow problem, it holds according to Proposition 3.6.19

$$\sum_{i \in N_S^W} v_i = \sum_{t \in \mathbb{N}} \sum_{i \in N_S^W} v_i^t - \sum_{i \in N_S^W} P_1(\tilde{\beta}^{-1}(i)) = \sum_{s \in S \setminus W} v^{d^\infty}(s) \cdot P_1(s).$$

The last equation followed from the previous equation 3.21.

\Leftarrow Assume a feasible solution $\{x_{i,j}, (i,j) \in \tilde{E}, i \in N_S\}$, $\{v_i, i \in N_S^W \cup N_S^L\}$ of the linear program 3.12. Then, from Theorem 3.6.21, a decision rule d of the SSO-MDP can be constructed by

$$\mathbb{P}\{d(\tilde{\beta}^{-1}(i)) = \tilde{\beta}^{-1}(j)\} := \begin{cases} \frac{x_{i,j}}{\sum_{k:(k,i) \in \tilde{E}} x_{k,i}} & \text{for all } (i,j) \in \tilde{E}, \quad i \in N_S \text{ s.t.} \\ & \sum_{k:(k,i) \in \tilde{E}} x_{k,i} > 0, \\ q_{d(\tilde{\beta}^{-1}(i))}(\tilde{\beta}^{-1}(j)) \text{ arbitrary s.t.} & \\ \sum_{\tilde{\beta}^{-1}(j) \in A_{d(\tilde{\beta}^{-1}(i))}} q_{d(\tilde{\beta}^{-1}(i))}(\tilde{\beta}^{-1}(j)) = 1, & \text{else} \\ q_{d(\tilde{\beta}^{-1}(i))}(\tilde{\beta}^{-1}(j)) \geq 0, & \end{cases}$$

such that

$$\sum_{s \in S \setminus W} P_1(s) v^{d^\infty}(s) = \sum_{i \in N_S^W} v_i$$

holds. ■

Finally, it can be conclude that the static maximum flow problem can be used to determine an optimal policy of an SSO-MDP:

Theorem 3.6.23 (Optimal stationary policy is characterized by static maximum flow formulation):
Assume the flow network $(\tilde{N}, \tilde{E}, \tilde{\beta}, \alpha)$ of an associated graph of an SSO-MDP and an initial distribution P_1 of the starting state.

Then, an optimal solution of the static maximum flow problem 3.12 characterizes an optimal stationary policy d^∞ of the SSO-MDP.

PROOF. Assume an optimal solution $\{x_{i,j}, (i,j) \in \tilde{E}, i \in N_S\}, \{v_i, i \in N_S^W \cup N_S^L\}$ of the static maximum flow problem. After Theorem 3.6.22, a stationary policy d^∞ of the SSO-MDP can be constructed such that

$$\sum_{s \in S \setminus W} v^{d^\infty}(s) \cdot P_1(s) = \sum_{i \in N_S^W} v_i.$$

Assume d^∞ is not an optimal policy of the SSO-MDP. Since there exists an optimal stationary policy in an SSO-MDP, there must be another stationary policy \tilde{d}^∞ with

$$\sum_{s \in S} v^{\tilde{d}^\infty}(s) \cdot P_1(s) > \sum_{s \in S} v^{d^\infty}(s) \cdot P_1(s).$$

Due to the absorbing property of all states $s \in W$, it holds

$$\sum_{s \in W} v^{d^\infty}(s) \cdot P_1(s) = \sum_{s \in W} P_1(s)$$

for all stationary policies $d^\infty \in \Pi^{MR}$. So, it can be concluded that

$$\sum_{s \in S \setminus W} v^{\tilde{d}^\infty}(s) \cdot P_1(s) > \sum_{s \in S \setminus W} v^{d^\infty}(s) \cdot P_1(s)$$

must hold. However after Theorem 3.6.22, a feasible solution $\tilde{x}_{i,j}, \forall (i,j) \in \tilde{E}, i \in N_S, \tilde{v}_i, \forall i \in N_S^W \cup N_S^L$ of linear program 3.12 can be found with

$$\sum_{i \in N_S^W} \tilde{v}_i > \sum_{i \in N_S^W} v_i.$$

This is a contradiction to the assumption that $x_{i,j}, v_i$ is an optimal solution of the linear program 3.12. So, d^∞ must be an optimal policy for the SSO-MDP. ■

The result of the last theorem is already mentioned in Subsection 3.5.2. However, the proof of Theorem 3.6.23 relies only on the results of this section which is why the theorem is included a second time in this thesis.

The next example revisits the SSO-MDP of Example 3.1 and illustrates the relation between the time-dependent flow variables and the found static solution of Example 3.1.

Example 3.8:

This example illustrates the relationship of the time-dependent flow and the static flow variables in the SSO-MDP of Example 3.1. Let

$$x_a^t := x_{i,j}^t \text{ with } \tilde{\beta}(i) = A^{-1}(a), \tilde{\beta}(j) = a$$

be the time-dependent flow variable for choosing action a at time t . And

$$x_{a,s}^t := x_{i,j}^t \text{ with } \tilde{\beta}(i) = a, \tilde{\beta}(j) = s$$

the time-dependent flow variable for a transition to a state s after having chosen action a at time t . As shown by Proposition 3.6.19 and Proposition 3.6.20, the relation

$$x_a = \sum_{t=1}^{\infty} x_a^t$$

is valid. The proof of Proposition 3.6.20 can be illustrated on the circle of this example. Consider the static flow variable x_{a_3} of the circle $s_2 - a_3 - s_2$. The probability of staying in this circle is 0.5. The only incoming flows to this circle that are unequal to zero are x_{a_1,s_2}^2 and x_{a_2,s_2}^2 . So, we have

$$\begin{aligned} x_{a_3} &= \sum_{t=1}^{\infty} x_{a_3}^t \\ &= 0 + 0 + \sum_{t=3}^{\infty} x_{a_3}^t \\ &= x_{a_3}^3 + 0.5 \cdot x_{a_3}^3 + 0.5^2 \cdot x_{a_3}^3 + \dots \\ &= x_{a_3}^3 \sum_{i=0}^{\infty} (0.5)^i \\ &= \frac{x_{a_3}^3}{1 - 0.5} = 2x_{a_3}^3 = 2 \cdot (q_{d(s_2)}(a_3) \cdot (x_{a_1,s_2}^2 + x_{a_2,s_2}^2)). \end{aligned}$$

As in the optimal solution $q_{d(s_2)}(a_3)$ is zero, we have $x_{a_3} = 0$ and $x_{a_3}^t = 0$ for all $t \in \mathbb{N}$. *

The main result of this Subsection is the static maximum flow formulation 3.12 together with Theorem 3.6.23 which proves the linear program can be solved for finding an optimal decision rule of the SSO-MDP.

3.7 Transforming SSO-MDPs

Value iteration applied to general infinite-horizon MDPs under the total expected reward criterion may take an infinite number of iterations (Littman, Dean, and Kaelbling, 1995). However, if the underlying graph of the SSO-MDP is acyclic, it can be shown that value iteration terminates after at most n steps at the optimal value function (Bertsekas, 2001, Sec. 2.2.1).

This motivates to develop an algorithm that transforms any network associated with an SSO-MDP in an acyclic network. Bertsekas explains in (Bertsekas, 2001, Sec. 2.2.2) how self-transitions can be eliminated. The algorithm presented in this section is a generalization to arbitrary circles that may occur in the graph of an SSO-MDP. Furthermore, Bertsekas considered a graph of only states where the transition probabilities are fixed to a certain policy. In the algorithm presented in this section the graph contains nodes related to actions and the resulting graph can be used to determine an optimal deterministic and stationary policy.

Assumption 3.2.1 of SSO-MDPs leads to some useful properties of a graph $G = (N, E, \beta)$ associated with an SSO-MDP. A definition of a graph associated with an SSO-MDP is given in Definition 3.6.1. In an associated graph of an SSO-MDP, there exist only edges from states to actions and visa versa. We have already seen that the graph is bipartite and splits into two disjoint set of nodes N_A and N_S which are called action nodes and state nodes, see Subsection 3.6.1. The graph extended by edge weights α according to Definition 3.6.3 is an associated flow network of an SSO-MDP.

An associated graph of an SSO-MDP may contain circles. A circle is a path in the graph, i.e., a sequence of edges in the graph, where the first and the last node are identical and no other node appears twice in the path. Remember, a circle C is denoted by $C = (v_1, \dots, v_k)$, where (v_{i-1}, v_i) is an edge in E and all nodes are distinct except $v_k = v_1$, see Definition 2.4.4. The set of edges contained in a circle is denoted by \mathcal{E}_C . Sometimes circles are also called elementary circuits. There exist several algorithms for finding all elementary circles in a directed connected graph. For instance, Johnson presents in Johnson (1975) an algorithm that finds all elementary circuits of a directed graph in linear time in terms of the number of nodes and edges. Most of the algorithms for finding circles rely on a depth-first search.

Before presenting an algorithm that transforms a network of an SSO-MDP in an acyclic network, another preliminary consideration is made. It can be shown that every node that has an incoming edge from a node not in the circle, must be a state node.

Proposition 3.7.1:

Let C be a circle in the associated graph G of an SSO-MDP. Let $v_i \in C$ be a node with an incoming flow from a node not in C , i.e., $|\delta_{in}(v_i) \cap \delta_{in}(C)| > 0$, then $v_i \in N_S$ holds.

PROOF. Assume $C = (v_1, \dots, v_k)$. Furthermore, let $v_i \in N_A$ be a node that corresponds to an action of the SSO-MDP. Assume that $|\delta_{in}(v_i) \cap \delta_{in}(C)| > 0$ holds. Let e be an edge in $\delta_{in}(v_i) \cap \delta_{in}(C)$. Then $start(e)$ must be a state node since G is bipartite, and there exists only edges between state and action nodes. From $e \in \delta_{in}(C)$, we can follow that $start(e) \notin C$ and $start(e)$ can not be equal to $v_{i-1} \in C$. But, $v_{i-1} \in C$ is another state node that is connected to v_i . This is a contradiction to our assumption of unique actions. ■

Algorithm 6 transforms a network of an SSO-MDP in an acyclic network. In the subsequent theorem, the correctness of that algorithm is proved and a relation between deterministic decision rules in the SSO-MDP derived from the transformed network and deterministic decision rules in the original SSO-MDP is specified afterwards.

For each circle C , the algorithm iterates over all edges that are outgoing edges of a node in C and do not lie in the set of edges \mathcal{E}_C contained in C . In concrete terms, this $\delta_-(C)$ is defined as

$$\delta_-(C) := \{e \in E \mid \exists v \in C : e \in \delta_{out}(v) \wedge e \notin \mathcal{E}_C\}.$$

Observe, that $\delta_{-}(C) \neq \delta_{out}(C)$ since $\delta_{-}(C)$ may contain an edge from an action node to a state node which may be both in C . For example, assume $C = (v_{s_1}, v_{a_1}, v_{s_2}, v_{a_2}, v_{s_1})$, which is a circle, and assume there exists an edge $e = (v_{a_1}, v_{s_1})$. Then, $e \in \delta_{-}(C)$ holds, but $e \notin \delta_{out}(C)$.

Algorithm 6 gets as input data a network obtained from an SSO-MDP and a set of circles. The set of circles should not contain the artificial circles at the absorbing nodes that belong to states in $W \cup L$. The algorithm iterates over all circles, and treats every circle separately. Thereby, for each node v_i of a circle C that has an incoming edge from a node which is not in C the following transformation is done: Each outgoing edge of a circle node that is not in the circle path, i.e., each edge in $\delta_{-}(C)$, is replaced by a new edge starting directly at v_i . Depending on the type of the end node of the outgoing edge, a different treatment is carried out:

- If there exists an end node that is a state node, an artificial action node \tilde{v}_i is inserted. This artificial action node will later correspond to a deterministic decision rule that chooses all actions of the circle. The weight of the new edge from the inserted action node to the end of the original outgoing edge is set to the cumulated probability of reaching that end of the outgoing edge. Figure 3.8 illustrates this treatment.
- If the end node of the outgoing edge of the circle is an action node, a copy of that node is made and an edge from v_i to the copied action node is added. The transition probabilities outgoing from that copied action node are multiplied by the probability of the path from v_i to the copied action node. Additional edges from the copied action node to every state that is adjacent to an outgoing edge of the path from v_i to the copied action node are generated. So, the copied action node corresponds to a deterministic decision rule that chooses all actions on that path from v_i to the copied action node. This situation is illustrated in Figure 3.9. If the considered action node is chosen with certainty, which is the case in a deterministic decision rule, there does not remain any flow in the circle. So, the cumulated probability of staying in the circle is not needed.

For calculating the cumulated transition probabilities of the artificial action node \tilde{v}_i , the limit of the geometric series is used. Due to our Assumption 3.2.1 of SSO-MDPs, the probability p , calculated by the loop in line 4, is always strictly smaller than 1. Therefore, the limit of the infinite geometric series is well defined and equals $\frac{1}{1-p}$. Another implication of Assumption 3.2.1 is that $\delta_{-}(C) \neq \emptyset$. So, we do not

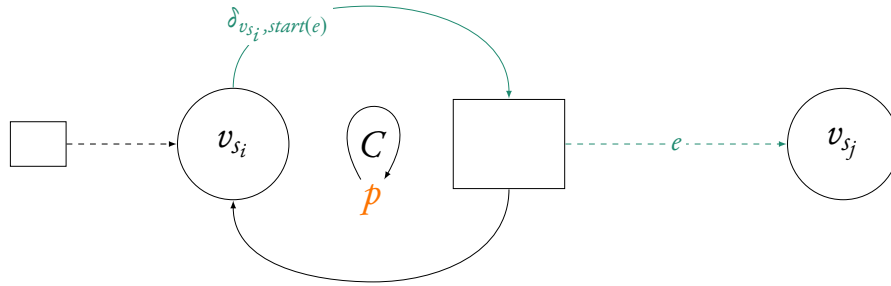
need to handle the case $\delta_-(C) = \emptyset$ in the algorithm.

Algorithm 6: Acyclic Transformation

Data: $G = (N, E, \beta)$ with edge weights α associated to an SSO-MDP;
 Ψ set of all circles with $\forall v \in C, C \in \Psi : v \notin L \cup W$
Result: $G' = (N', E', \beta')$ with edge weights α' acyclic

- 1 Set $N' \leftarrow N, E' \leftarrow E, \beta' \leftarrow \beta, \alpha' \leftarrow \alpha$;
- 2 for all $C = (v_1, \dots, v_k) \in \Psi$ do
 - 3 $p \leftarrow 0$;
 - 4 for all $e \in \mathcal{E}_C$ do
 - 5 $p \leftarrow p \cdot \alpha(e)$;
 - 6 end
 - 7 compute $\gamma \leftarrow \frac{1}{1-p}$;
 - 8 for all $v_i \in C$ with $|\delta_{in}(v_i) \cap \delta_{in}(C)| > 0$ do
 - 9 if $\exists e \in E : e \in \delta_-(C)$ with $\text{end}(e) \in N_S$ then
 - 10 add an action node \tilde{v}_i to N' ;
 - 11 set edge $\tilde{e} \leftarrow (v_i, \tilde{v}_i), \alpha'(\tilde{e}) \leftarrow 1$;
 - 12 insert edge \tilde{e} in E' ;
 - 13 set $\beta'^{-1}(\tilde{v}_i) = \{a \in \mathcal{A} \mid \beta(a) \in C\}$
 - 14 end
 - 15 for $e \in \delta_-(C), e \in E$ do
 - 16 if $\text{end}(e) \in N_S$ then
 - 17 determine $\mathcal{E}_{v_i, \text{start}(e)}$ with edges in \mathcal{E}_C ;
 - 18 set edge $\tilde{e} \leftarrow (v_i, \text{end}(e)), \alpha'(\tilde{e}) \leftarrow \gamma \cdot \alpha(\mathcal{E}_{v_i, \text{start}(e)}) \cdot \alpha(e)$;
 - 19 insert edge \tilde{e} in E' ;
 - 20 else
 - 21 add action node \tilde{a}_i to N' ;
 - 22 insert edge $\tilde{e} \leftarrow (v_i, \tilde{a}_i), \alpha'(\tilde{e}) \leftarrow 1$ in E' ;
 - 23 determine $\mathcal{E} \leftarrow \mathcal{E}_{v_i, \text{end}(e)}$ in E with $\mathcal{E}_{v_i, \text{start}(e)}$ edges in \mathcal{E}_C ;
 - 24 set $\beta'^{-1}(\tilde{a}_i) = \{a \in \mathcal{A} \mid \beta(a) \in \mathcal{V}(\mathcal{E}_{v_i, \text{end}(e)})\}$;
 - 25 for $\tilde{e} \in \delta_-(\mathcal{E}), \tilde{e} \in E$ with $\text{end}(\tilde{e}) \in N_S$ do
 - 26 determine $\mathcal{E}_{\tilde{e}} \leftarrow \mathcal{E}_{v_i, \text{start}(\tilde{e})}$ with edges in \mathcal{E} ;
 - 27 set edge $\tilde{e} \leftarrow (\tilde{a}_i, \text{end}(\tilde{e})), \alpha'(\tilde{e}) \leftarrow \alpha(\mathcal{E}_{\tilde{e}}) \cdot \alpha(\tilde{e})$;
 - 28 insert edge \tilde{e} in E' ;
 - 29 end
 - 30 end
 - 31 end
 - 32 end
 - 33 for $e \in \delta_-(C), e \in E$ do
 - 34 if $\text{end}(e) \in N_A$ then
 - 35 for $\tilde{e} \in \delta_{out}(\text{end}(e))$ do
 - 36 remove \tilde{e} from E' ;
 - 37 end
 - 38 remove $\text{end}(e)$ from N', β' ;
 - 39 end
 - 40 remove e from E'
 - 41 end
 - 42 remove all $e \in \mathcal{E}_C$ with $e \in E$ from E' ;
 - 43 remove all $v \in C$ with $|\delta_{in}(v)| = 0$ from N', β' ;
 - 44 end

$$G = (N, E, \beta):$$



$$G' = (N', E', \beta')$$

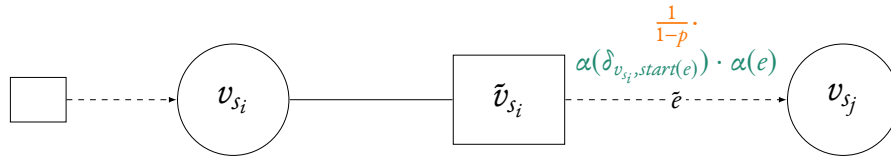


Figure 3.8: Illustration of Algorithm 6 for $e \in E : e \in \delta_-(C)$ with $end(e) \in N_S$

Note that the returned mapping β' is no bijection between the original set of states and actions of the SSO-MDP, and the new nodes N' . This is a requirement of Definition 3.6.1 of a graph associated with an SSO-MDP. However, if one summarizes actions in the SSO-MDP to new cumulated actions as in the algorithm, β' is again a bijection. In Theorem 3.7.4, where the acyclic SSO-MDP is needed, an action set $\mathcal{A}_{acyclic}$ is defined that consists of action sets such that β' is a bijection to the nodes in the transformed acyclic graph.

In the circle set Ψ , there may be circles that have common nodes. After one circle has been processed by the algorithm, all circles that had nodes in common with this circle are still contained in G' . But their notation must be updated to the new nodes in N that occurred from the processing of the previous circles.

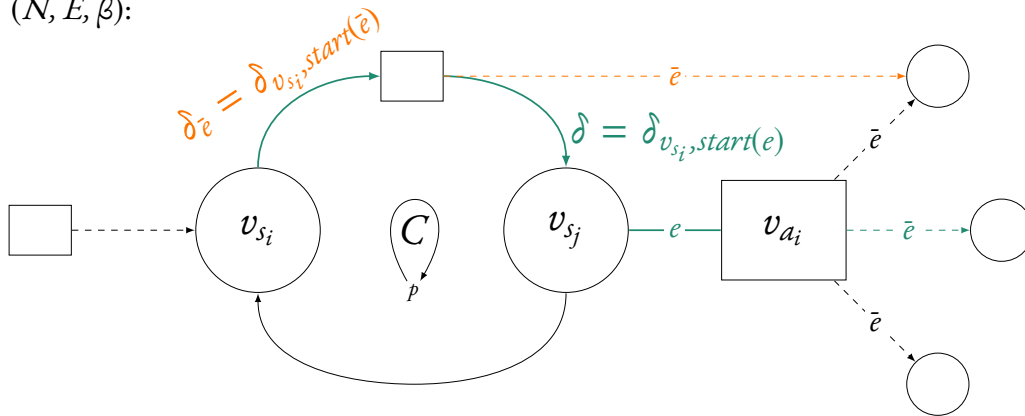
We start showing that the returned network is indeed acyclic. Furthermore, we show that the structure in terms of states, actions, rewards and transitions suites to an SSO-MDP.

Proposition 3.7.2:

Let $G' = (N', E', \beta')$ with edge weights α' be the network returned from Algorithm 6. Then:

1. G' is a connected and acyclic directed network.
2. $(N', E', \beta', \alpha')$ is an associated flow network of an SSO-MDP according to Definition 3.6.3 with starting state s_1 , winning states W and losing states L from the original SSO-MDP.

$G = (N, E, \beta)$:



$G' = (N', E', \beta')$:

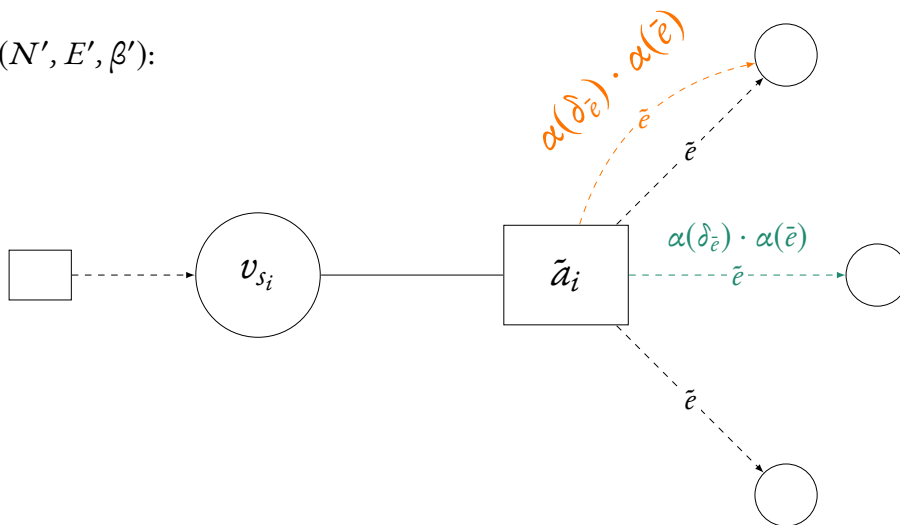


Figure 3.9: Illustration of Algorithm 6 for $e \in E : e \in \delta_{-}(C)$ with $end(e) \in N_A$

PROOF. I. The input graph G of Algorithm 6 is connected since it is an associated graph of an SSO-MDP. We show that the resulting graph $G' = (N', E', \beta')$ is a connected graph. At the initialization, the set of nodes N' is set to N and the set of edges E' to E . Let $C \in \Psi$ be an arbitrary circle. In lines 10 and 21 new nodes are added to N' . However, these new nodes are both, in the subsequent step, connected by an edge \bar{e} to a node $v_i \in C$ which has an incoming edge. Before line 33, where edges and nodes are removed, G' is connected. In line 33, in a for loop, all edges starting in a node of C that are not in ξ_C are removed from E' . We show that the end nodes of these outgoing edges are still connected to other nodes of the graph. If the end node $end(e)$ is a state node, an edge from the artificial action node \bar{v}_i to the end node $end(e)$ exists. If $end(e)$ is an action node, all outgoing edges \bar{e} of that action node are replaced by edges from the artificial action node \bar{a}_i to $end(\bar{e})$. So, we have a connection from v_i over \bar{a}_i to all successors of $end(e)$. Removing $e \in \delta_-(C)$, $end(e) \in N_A$ and all $\bar{e} \in \delta_{out}(end(e))$ still yields a connected graph. Since in line 43 only nodes with no incoming edge from outside the circle are removed from the graph, all v_i with an incoming edge will stay in N' . Therefore, the artificial action nodes \bar{v}_i and \bar{a}_i remain connected to the graph. Finally, removing edges that are in ξ_C do not lead to a disconnected graph since either $v_i \in C$ has an incoming edge from outside the circle and is therefore connected to the rest of the graph, or it will be removed from N' .

Ψ is a complete enumeration of all circles (elementary circuits) except the artificial circles at the absorbing state nodes that belong to $W \cup L$. Roughly speaking, Algorithm 6 replaces all edges inside a circle by new paths that connect each node with an incoming edge from a non-circle node with all nodes that were adjacent to the circle. Since all edges with both ends in C are removed from E' , C can not be a circle any more. The artificial action nodes generated by the algorithm connects a circle node with a non-circle node. Furthermore, the artificial action nodes are newly generated for each node v_i with an incoming edge. Obviously, after handling a single circle by Algorithm 6, the handled circle is removed.

Next, it is shown that no new circles may occur after the transformation. Assume Algorithm 6 has processed all circles. Let C' be a new circle that has occurred after applying the algorithm. If C' was contained in Ψ before, it would have been removed. So, C' can not have been in Ψ . Since C' was not in Ψ , in $\xi_{C'}$ there must be at least one new edge generated by the algorithm. Algorithm 6 generates only new edges that start or end at an artificial action node. These action nodes have a unique predecessor, a state node v_i with an incoming edge from a node not contained in the circle. So, a circle C' occurred from a newly generated edge, must contain also an artificial action node and its predecessor which is a state node v_i that was a part of a circle $C \in \Psi$.

Suppose, $C' = (v_1, \dots, v_k, v_1)$ is a new circle evolved after the application of the algorithm. Assume C' has nodes v_{p_i} in common with C_i , $i = 1, \dots, l$ and $C_1, \dots, C_l \in \Psi$ is a subset of circles. So w.l.o.g., C' must be of the form

$$C' = (v_1, \dots, v_{p_1}, \bar{v}_{p_1}, \dots, v_{p_l}, \bar{v}_{p_l}, \dots, v_k, v_1),$$

where \bar{v}_{p_i} is an artificial action node generated by the algorithm. All artificial action nodes connect only nodes that were a priori connected by a path in G . We can replace all v_{p_i} plus the artificial action nodes \bar{v}_{p_i} in C' by a path existing in G . Or more precisely, if \bar{v}_{p_i} was an artificial action node generated in line 10, we can replace (v_{p_i}, \bar{v}_{p_i}) by a path in ξ_C plus an edge in $\delta_-(C)$.

And if \tilde{v}_{p_i} was an artificial action node generated in line 21, we can replace it by a path in \mathcal{g}_C plus an edge e in $\delta_{-}(C)$ and an edge $\bar{e} \in \delta_{out}(end(e))$. So, we can conclude that the following path must have existed a priori in G :

$$(v_1, \dots, v_{p_i-1}, \xi(v_{p_i}, v_{p_i+1}), \dots, v_{p_i-1}, \xi(v_{p_i}, v_{p_i+1}), \dots, v_k, v_1).$$

This is a circle with nodes in N , edges in E . This circle or at least its elementary sub-circles must have been contained in Ψ and therefore must have been eliminated by the algorithm. We have seen that there can not arise new circles from the transformation of Algorithm 6 and the transformed graph G' is acyclic.

2. In order to prove that $(N', E', \beta', \alpha')$ is a flow network associated with an SSO-MDP, we have to show that G' contains only edges from states to action nodes and vice versa. Furthermore, the weight of every edge from a state node to an action node should be 1 and the weights of all edges outgoing from an action node to connected state nodes should define a probability distribution.

Let $C \in \Psi$ be a circle and $N^C = N_A^C \cup N_S^C$ be the nodes of that circle, which can be distinguished in state nodes N_S^C and action nodes N_A^C . In the next step, it is shown that after processing C , the weights of the edges E' fulfill the mentioned requirements. We know from Proposition 3.7.1 that all nodes with an incoming edge from a node not in C must be a state node. Let N_{S-}^C be all state nodes of C with an incoming edge from nodes not in C . All new generated edges in Algorithm 3.2.1 start either at a state node $v_i \in N_{S-}^C$ or at an artificial action node \tilde{v}_i or \tilde{a}_i . Observe, that at the end of the algorithm all $v \in N^C \setminus N_{S-}^C$, all $e \in \mathcal{g}_C \cup \delta_{-}(C)$ and all nodes with $v \in N_A^C$ with $(\cdot, v) \in \delta_{-}(C)$ and its edges $\delta_{out}(v)$ are removed. We can summarize that after processing C , modified edges occur only in $\delta_{out}(v_i)$ with $v_i \in N_{S-}^C$, and $\delta_{out}(\tilde{v}_i) \cup \delta_{out}(\tilde{a}_i)$. All other nodes are either removed or stayed unchanged.

We first consider edges in $\delta_{out}(v_i)$, $v_i \in N_{S-}^C$. These are edges generated in line 11 or line 22 which go to an artificial action node and have a weight of 1.

Next, it is verified that the edge weights of edges in $\delta_{out}(\tilde{v}_i)$ specify a probability distribution outgoing from that action node. The algorithm generates an artificial action node \tilde{v}_i when processing $v_i \in N_{S-}^C$. For easier reading, in the following an action node of C is denoted by v_{a_i} , a state node by v_{s_i} . Furthermore, any path $\xi_{v,w}$ between nodes v and w using only edges in C is denoted by $\xi_{v,w}^C$. Figure 3.8 illustrates this situation and the transformation that is carried out. Let

$$C = (v_{s_1}, v_{a_1}, \dots, v_{s_k}, v_{a_k}, v_{s_1}).$$

Without loss of generality, we can assume that $v_i \in N_{S-}^C$ is the first node v_{s_1} of C . Then,

$$\begin{aligned} & \sum_{e \in \delta_{out}(\tilde{v}_i)} \alpha'(e) \\ = & \sum_{e \in \delta_{-}(C) \text{ with } end(e) \in N_S} \gamma \cdot \alpha(\xi_{v_i, start(e)}^C) \cdot \alpha(e) \\ = & \sum_{e \in \delta_{-}(C) \text{ with } start(e) \in N_A^C} \gamma \cdot \alpha(\xi_{v_i, start(e)}^C) \cdot \alpha(e) \end{aligned}$$

$$\begin{aligned}
&= \gamma \sum_{v_{a_l} \in C} \alpha(\xi_{v_{s_1}, v_{a_l}}^C) \sum_{e \in \delta_{out}(v_{a_l}) \cap \delta_-(C)} \alpha(e) \\
&= \gamma \sum_{v_{a_l} \in C} \alpha(\xi_{v_{s_1}, v_{a_l}}^C) (1 - \alpha(\xi_{v_{a_l}, v_{s_{l+1}}}^C)) \\
&= \gamma \sum_{v_{a_l} \in C} \left(\alpha(\xi_{v_{s_1}, v_{a_l}}^C) - \alpha(\xi_{v_{s_1}, v_{s_{l+1}}}^C) \right) \\
&\stackrel{\alpha(v_{s_l}, v_{a_l})=1}{=} \gamma \sum_{v_{a_l} \in C} \left(\alpha(\xi_{v_{s_1}, v_{a_l}}^C) - \alpha(\xi_{v_{s_1}, v_{a_{l+1}}}^C) \right) \\
&\stackrel{\text{telescope sum}}{=} \gamma \left(\alpha(\xi_{v_{s_1}, v_{a_1}}^C) - \alpha(\xi_{v_{s_1}, v_{a_k}}^C) \alpha(\xi_{v_{a_k}, v_{a_1}}^C) \right) \\
&= \gamma \left(\alpha(\xi_{v_{s_1}, v_{a_1}}^C) - \alpha(\xi_{v_{s_1}, v_{a_k}}^C) \alpha(\xi_{v_{a_k}, v_{s_1}}^C) \right) \\
&= \gamma \cdot (1 - p) = \frac{1-p}{1-p} = 1.
\end{aligned}$$

In the sixth equation, it is used that G is a graph associated to an SSO-MDP. Therefore, we know that each edge from a state node to an action node has weight 1. As a weight of a path is the product of all weights on the path, we can follow that $\alpha(\xi_{v_{s_l}, v_{s_{l+1}}}^C) = \alpha(\xi_{v_{s_l}, v_{a_{l+1}}}^C)$. At the end, the formula of γ from the algorithm is used and the fact that p was calculated in the algorithm by multiplying all weights of the edges on the circle which equals $\alpha(\xi_{v_{s_1}, v_{s_1}}^C)$.

Finally, it is shown that the edges in $\delta_{out}(\tilde{a}_i)$ specify a probability distribution. Let again $C = (v_{s_1}, v_{a_1}, \dots, v_{s_k}, v_{a_k}, v_{s_1})$ and assume again that the currently processed $v_i \in N_{S-}^C$ is the first node v_{s_1} of C . Let $e \in \delta_-(C)$ be the edge that was considered, when generating \tilde{a}_i in the algorithm (line 21). Assume $e = (v_{s_j}, v_{a_i})$, where v_{s_j} is a node in C and $v_{a_i} \notin C$. Furthermore, let $\xi = \xi_{v_i, end(e)}$ as in the algorithm, where $\xi_{v_i, start(e)}$ is a path in C as specified in the algorithm. Figure 3.9 illustrates that situation and the transformation that is carried out by the algorithm.

The proof is divided into parts to make it better readable. The idea is to go backwards from v_{a_i} to v_{s_1} along the path by finding probability distributions that sum up to 1. In the first step, we get rid of the final node v_{a_i} :

$$\begin{aligned}
&\sum_{\tilde{e} \in \delta_{out}(\tilde{a}_i)} \alpha'(\tilde{e}) \\
\text{Alg. line 27} \quad &= \sum_{\tilde{e} \in \delta_-(\xi), end(\tilde{e}) \in N_S} \alpha(\xi_{\tilde{e}}) \cdot \alpha(\tilde{e}) \\
&= \sum_{\tilde{e} \in \delta_-(\xi_{v_{s_1}, v_{a_i}}), end(\tilde{e}) \in N_S} \alpha(\xi_{\tilde{e}}) \cdot \alpha(\tilde{e}) \\
&= \sum_{\tilde{e} \in \delta_-(\xi_{v_{s_1}, v_{a_i}}) \cap \delta_{out}(v_{a_i})} \alpha(\xi_{v_{s_1}, end(\tilde{e})}) \\
&\quad + \sum_{\tilde{e} \in \delta_-(\xi_{v_{s_1}, v_{a_i}}) \setminus \delta_{out}(v_{a_i}), end(\tilde{e}) \in N_S} \alpha(\xi_{v_{s_1}, end(\tilde{e})}) \tag{3.22}
\end{aligned}$$

$$= \sum_{\bar{e} \in \delta_{out}(v_{a_i})} \alpha(\xi_{v_{s_1}, v_{a_i}}) \cdot \alpha(\bar{e}) \quad (3.23)$$

$$+ \sum_{\bar{e} \in \delta_{-}(\xi_{v_{s_1}, v_{a_i}}) \setminus \delta_{out}(v_{a_i}), \text{end}(\bar{e}) \in N_S} \alpha(\xi_{v_{s_1}, \text{end}(\bar{e})})$$

$$= \alpha(\xi_{v_{s_1}, v_{a_i}}) + \sum_{\bar{e} \in \delta_{-}(\xi_{v_{s_1}, v_{s_j}}), \text{end}(\bar{e}) \in N_S} \alpha(\xi_{v_{s_1}, \text{end}(\bar{e})})$$

$$= \alpha(\xi_{v_{s_1}, v_{s_j}}^C) + \sum_{\bar{e} \in \delta_{-}(\xi_{v_{s_1}, v_{s_j}}^C), \text{end}(\bar{e}) \in N_S} \alpha(\xi_{v_{s_1}, \text{end}(\bar{e})}) \quad (3.24)$$

When in the last equations a path $\xi_{v_{s_1}, v_{a_i}}$ is used, it is assumed to be an extension of the path $\xi_{v_{s_1}, v_{s_j}}^C$ by (v_{s_j}, v_{a_i}) . Furthermore, $\xi_{v_{s_1}, \text{end}(\bar{e})}$ is also assumed to be a path in C except for edge \bar{e} . In Equation 3.22, the edge set $\delta_{-}(\xi_{v_{s_1}, v_{a_i}})$ is divided in a set of edges that start in v_{a_i} and a set of edges which do not start in v_{a_i} . Since G is bipartite, all edges in $\delta_{out}(v_{a_i})$ satisfy $\text{end}(\bar{e}) \in N_S$. In Equation 3.23, it is used that all outgoing edges of $\delta_{out}(v_{a_i})$ are included in $\{\bar{e} \in E \mid \delta_{-}(\xi_{v_{s_1}, v_{a_i}})\}$. This holds since all e in $\delta_{out}(v_{a_i})$ are incident to node v_{a_i} which is a node of $\xi_{v_{s_1}, v_{a_i}}$ and at the same time $\xi_{v_{s_1}, v_{a_i}}$ does not contain edges of $\delta_{out}(v_{a_i})$. In Equation 3.24, it is used that $\alpha(v_{s_j}, v_{a_i}) = 1$ since it is a state-action edge in an SSO-MDP.

We arrived at a sum over edges that are incident with edges of the path $\xi_{v_{s_1}, v_{s_j}}^C$ but are not included in that path. Observe that the path $\xi_{v_{s_1}, v_{s_j}}^C$ is a part of ξ_C . In the next step, the sum over the edges belonging to the last state and action is simplified. So, the considered path has been shortened by the last action and the last state.

$$\alpha(\xi_{v_{s_1}, v_{s_j}}^C) + \sum_{\bar{e} \in \delta_{-}(\xi_{v_{s_1}, v_{s_j}}^C), \text{end}(\bar{e}) \in N_S} \alpha(\xi_{v_{s_1}, \text{end}(\bar{e})})$$

$$= \alpha(\xi_{v_{s_1}, v_{s_j}}^C) + \sum_{\bar{e} \in \delta_{-}(\xi_{v_{s_1}, v_{s_j}}^C) \cap \delta_{out}(v_{a_{j-1}})} \alpha(\xi_{v_{s_1}, \text{end}(\bar{e})})$$

$$+ \sum_{\bar{e} \in \delta_{-}(\xi_{v_{s_1}, v_{s_j}}^C) \setminus \delta_{out}(v_{a_{j-1}}), \text{end}(\bar{e}) \in N_S} \alpha(\xi_{v_{s_1}, \text{end}(\bar{e})})$$

$$= \alpha(\xi_{v_{s_1}, v_{a_{j-1}}}^C) \alpha(\xi_{v_{a_{j-1}}, v_{s_j}}^C)$$

$$+ \sum_{\bar{e} \in \delta_{out}(v_{a_{j-1}}) \setminus (v_{a_{j-1}}, v_{s_j})} \alpha(\xi_{v_{s_1}, v_{a_{j-1}}}^C) \alpha(\bar{e}) \quad (3.25)$$

$$+ \sum_{\bar{e} \in \delta_{-}(\xi_{v_{s_1}, v_{a_{j-1}}}^C) \setminus \delta_{out}(v_{a_{j-1}}), \text{end}(\bar{e}) \in N_S} \alpha(\xi_{v_{s_1}, \text{end}(\bar{e})})$$

$$= \alpha(\xi_{v_{s_1}, v_{a_{j-1}}}^C) + \sum_{\bar{e} \in \delta_{-}(\xi_{v_{s_1}, v_{a_{j-1}}}^C) \setminus \delta_{out}(v_{a_{j-1}}), \text{end}(\bar{e}) \in N_S} \alpha(\xi_{v_{s_1}, \text{end}(\bar{e})})$$

$$= \alpha(\xi_{v_{s_1}, v_{s_{j-1}}}^C) + \sum_{\bar{e} \in \delta_{-}(\xi_{v_{s_1}, v_{s_{j-1}}}^C), \text{end}(\bar{e}) \in N_S} \alpha(\xi_{v_{s_1}, \text{end}(\bar{e})})$$

Again, when in the last equations a path $\xi_{v_{s_1}, \text{end}(\bar{e})}$ is used, it is assumed to be an extension of a path in C by the edge \bar{e} . The sum over all edges in $\delta_{-}(\xi_{v_{s_1}, v_{s_j}}^C)$ is again split up according to whether the edges are in $\delta_{\text{out}}(v_{a_{j-1}})$ or not. In Equation 3.25, it is used that $v_{a_{j-1}}$ is a node on the path $\xi_{v_{s_1}, v_{a_{j-1}}}^C$. So, all edges in $\delta_{\text{out}}(v_{a_{j-1}})$ are also in $\delta_{-}(\xi_{v_{s_1}, v_{a_{j-1}}}^C)$ except the edge $(v_{a_{j-1}}, v_{s_j})$, which is an edge of the circle.

This procedure is used until we arrive at v_{s_1} :

$$\begin{aligned}
&= \alpha(\xi_{v_{s_1}, v_{s_2}}^C) + \sum_{\bar{e} \in \delta_{-}(\xi_{v_{s_1}, v_{s_2}}^C), \text{end}(\bar{e}) \in N_S} \alpha(\xi_{v_{s_1}, \text{end}(\bar{e})}) \\
&= \alpha(\xi_{v_{s_1}, v_{a_1}}^C) \alpha(\xi_{v_{a_1}, v_{s_2}}^C) + \sum_{\bar{e} \in \delta_{\text{out}}(v_{a_1}) \setminus (v_{a_1}, v_{s_2})} \alpha(\xi_{v_{s_1}, v_{a_1}}^C) \alpha(\bar{e}) \\
&= \alpha(\xi_{v_{s_1}, v_{a_1}}^C) \\
&= 1.
\end{aligned}$$

It has been shown that at all artificial action nodes inserted by the algorithm the transition probabilities define a valid probability distribution and the requirements on the edge weights α are satisfied. \blacksquare

The following technical lemma shows a relation between the edge weights of G and G' . Namely, that the sum over the weights of all walks from a non-deleted state node of a circle through the circle to a state node adjacent to this circle remains the same under the transformation of Algorithm 6. From Lemma 3.7.3 and Proposition 3.7.2, it also follows that the SSO-MDP derived from G' satisfies Assumption 3.2.1 of SSO-MDPs.

Lemma 3.7.3:

Let $G = (N, E, \beta)$ with edge weights α be the associated network of an SSO-MDP and $G' = (N', E', \beta')$ with edge weights α' be the acyclic network computed from G and α by Algorithm 6. Let $C = (v_{s_1}, v_{a_1}, \dots, v_{s_k}, v_{a_k}, v_{s_1})$ be a circle of G , v_{s_i} a state node with an incoming flow from outside of C and N'_{s_i}, E'_{s_i} the sets of new nodes and edges generated by Algorithm 6 when processing v_{s_i} . Then, for all $v_s \in N_S$ with an edge $e \in \delta_{-}(C)$ and $\text{end}(e) = v_s$

$$\sum_{\omega_{v_{s_i}, v_s} \text{ with } \omega_{v_{s_i}, \text{start}(e)} \in C} \alpha(\omega_{v_{s_i}, v_s}) = \sum_{\omega_{v_{s_i}, v_s} \in E'_{s_i}} \alpha'(\omega_{v_{s_i}, v_s})$$

holds.

PROOF. Assume $v_s \in N_S$ with $\text{end}(e) = v_s$ and $e \in \delta_{-}(C)$. Since C is an elementary circle, there exists a unique path $\xi_{v_{s_i}, \text{start}(e)}$ in C . Furthermore, in the acyclic graph G' , there exists after construction only the path $(v_{s_i}, \tilde{v}_{s_i}, v_s)$ in E'_{s_i} that connects v_{s_i} with v_s .

$$\sum_{\omega_{v_{s_i}, v_s} \in E'_{s_i}} \alpha'(\omega_{v_{s_i}, v_s})$$

$$\begin{aligned}
&= \alpha'((v_{s_i}, \tilde{v}_{s_i}, v_s)) \\
&= \alpha'((v_{s_i}, \tilde{v}_{s_i})) \cdot \alpha'((\tilde{v}_{s_i}, v_s)) \\
&= 1 \cdot \gamma \cdot \alpha(\varrho_{v_{s_i}, \text{start}(e)}) \cdot \alpha(\text{start}(e), v_s) \tag{3.26} \\
&= \frac{1}{1-p} \alpha(\varrho_{v_{s_i}, v_s}) \\
&= \alpha(\varrho_{v_{s_i}, v_s}) \sum_{k=0}^{\infty} p^k \\
&= \alpha(\varrho_{v_{s_i}, v_s}) \sum_{k=0}^{\infty} \alpha(\varrho_{v_{s_i}, v_{s_i}})^k \tag{3.27} \\
&= \sum_{\omega_{v_{s_i}, v_s} \text{ with } \omega_{v_{s_i}, \text{start}(e)} \in C} \alpha(\omega_{v_{s_i}, v_s}).
\end{aligned}$$

In Equation 3.26, line 11 of the Algorithm 6 was used and in Equation 3.27, it was applied that p is the probability of staying in the circle C . ■

If a graph G and edge weights α associated with an SSO-MDP are transformed by Algorithm 6, the derived SSO-MDP from the outputted graph G' and edge weights α' is called the transformed acyclic SSO-MDP and denoted by SSO-MDP_{acyclic}. SSO-MDP_{acyclic} contain actions, states according to the new generated action nodes and the remaining state nodes. Define the state set as

$$S_{acyclic} := \{\beta'^{-1}(v_s) \mid v_s \in N'_S\},$$

which are just the states according to the remaining state nodes in N' . The new generated actions correspond to aggregated original actions. For this reason, $A_{acyclic}$ is defined as a set of action sets:

$$A_{acyclic} := \{\beta^{-1}(v_a) \mid v_a \in N'_A\}.$$

This is a set of action sets since e.g. $\beta^{-1}(\tilde{v}_i)$ is mapped by the algorithm to all actions of the processed circle.

As a final result, it is shown that every deterministic stationary policy d^∞ of an SSO-MDP is equivalent to a stationary deterministic policy d'^∞ of the transformed acyclic SSO-MDP_{acyclic} in the sense that it has the same winning probability.

Theorem 3.7.4 (Acyclic Transformation):

Let SSO-MDP_{acyclic} be the transformed acyclic SSO-MDP derived from the graph G' of Algorithm 6. Let d^∞ be a deterministic stationary policy of the SSO-MDP. Define a deterministic stationary policy d'^∞ of the SSO-MDP_{acyclic} as

$$d'(A') := 1, A' \in A_{acyclic} \Leftrightarrow d(a) = 1, \forall a \in A'.$$

Then, the value of d'^∞ in the SSO-MDP_{acyclic} equals the value of d^∞ in the original SSO-MDP.

PROOF. The set of losing states and winning state have not changed through the transformation by Algorithm 6. From Lemma 3.7.3, it follows that for two arbitrary states s_1, s_2 that are both in N_S and N'_S , the sum of all edge weights of walks from s_1 to s_2 remains unchanged in $\text{SSO-MDP}_{acyclic}$.

Decision rule d chooses the action nodes along a path if and only if d' chooses the action set that resulted from the transformation of that path. So, the sum of edge weights of walks to a winning state under a given policy stays unchanged. As this corresponds to the probability of reaching the winning state under this policy, it equals the value of the policies according to Proposition 3.4.1.

Observe that it is not possible to find a decision d' of the $\text{SSO-MDP}_{acyclic}$ that is equivalent to a truly randomized decision rule d of the SSO-MDP. This can be seen as follows: Assume d chooses all actions along the circle path with probability q and with probability $q - 1$ it leaves the circle. Let p again be the probability of staying in the circle if all actions along the circle are chosen with certainty. Then the probability of not leaving the circle under d equals

$$\sum_{k=1}^{\infty} (q \cdot p)^k = \frac{1}{1 - q \cdot p}.$$

Obviously this probability is not linear in q . In the transformed $\text{SSO-MDP}_{acyclic}$, there exists an action set in $\mathcal{A}_{acyclic}$ that contains all actions contained in the circle. The probability included in the transitions from that new generated actions is $\frac{1}{1-p}$ and corresponds to the probability of staying in the circle if all actions on the circle are chosen with certainty. Now $q \cdot \frac{1}{1-p}$ is obviously unequal to $\frac{1}{1-q \cdot p}$ for $q \neq 1$ and it can be seen that we can not define a decision rule d' as a randomization of deterministic actions in the $\text{SSO-MDP}_{acyclic}$. We would have to define new action nodes with different transition probabilities for every randomized decision rule d . As there exists infinitely many randomized decision rules such an approach would not be useful. However, this is no damage since we know that there exists an optimal stationary and deterministic decision rule in each MDP.

Corollary 3.7.5:

For every SSO-MDP there is a transformed SSO-MDP whose states have a topological order and which has the same value.

PROOF. It was shown that every directed graph belonging to an SSO-MDP can be transformed via Algorithm 6 in an acyclic directed graph. Every directed acyclic graph admits a topological sorting Jungnickel, 2008, Thm. 2.6.3. In Theorem 3.7.4, it was shown that for each deterministic stationary policy of the acyclic $\text{SSO-MDP}_{acyclic}$ there exists an equivalent deterministic stationary policy of the original SSO-MDP. As among the optimal policies there always exists an optimal stationary policy the values of both MDPs must be identical. ■

As already motivated at the beginning of this Subsection, the topological ordering can be used to show that value iteration applied to SSO-MDPs terminates after at most n steps at the optimal value function.

3.8 Further Extensions of SSO-MDPs

3.8.1 Randomized Strategies

In an MDP with a finite state and action space, there always exists a stationary deterministic optimal policy, see Section 2.3. Therefore, a randomized strategy in an SSO-MDP will never be strictly better than every deterministic strategy. However, in a sports game, there exist situations in which a randomized strategy should be strictly preferred over all pure strategies.

Consider, for example, beach volleyball: If each serve has an identical direction, the opponent team will after some time adapt themselves to that serve. Randomizing between some targets fields may be preferred such that the opponent team may not predict the direction of the serve. Physical exhausting actions are another example: For example in soccer, if a player should perform a sprinting action many times with only short or no breaks, he may be exhausted after a while. The sprinting actions will get slower and lose its desired effect. An insertion of a short break in form of a trot would help the player to regenerate. So, a randomization between sprints and trots may be preferred over sprinting all the time.

There are more examples of sports game situations where a randomized strategy should be strictly preferred. We want to propose an extension of SSO-MDPs, called *constraint SSO-MDPs*, that will achieve the desired result. Let A' be again the augmented action set consisting of state-action pairs $a = (s, \bar{a}) \in A'$.

Definition 3.8.1 (Constraint SSO-MDPs).

A *constraint SSO-MDP* is an SSO-MDP with lower bounds $l(a)$ or upper bounds $u(a)$ on some decisions $d(a)$, $a \in A'$. Let $C \subseteq A'$ be the subset of state-action pairs that are constrained, and l and u functions $C \rightarrow \mathbb{R}$ that map an action $a \in C$ to its lower bound $l(a)$ respectively upper bound $u(a)$.

To include those constraints on the decision rule d in the dual linear programming formulation of SSO-MDPs, we have to calculate constraints for the flow variables x_a with $a = (s, \bar{a})$ in the following way:

$$\begin{aligned} l(a) &\leq d(a) \leq u(a), \forall a \in C \\ \Leftrightarrow l(a) \sum_{\bar{a} \in A_s} x_a &\leq x_a \leq u(a) \sum_{\bar{a} \in A_s} x_a, \forall a = (s, \bar{a}) \in C. \end{aligned}$$

For actions $a = (s, \bar{a}) \in C$ with $\sum_{\bar{a} \in A_s} x_a > 0$, the equivalence holds by definition of a decision rule derived from a feasible solution of the dual linear program, see Definition 2.3.29. For actions $a = (s, \bar{a}) \in A'$ with $\sum_{\bar{a} \in A_s} x_a = 0$, the inequalities above simplify to $0 \leq x_a \leq 0 \Leftrightarrow x_a = 0$. Since $\sum_{\bar{a} \in A_s} x_a = 0$ if and only if $x_a = 0$ for all $a = (s, \bar{a}) \in A_s'$, this inequalities above does not change the feasibility set. Furthermore, in the case $\sum_{\bar{a} \in A_s} x_a = 0$ an arbitrary decision rule can be chosen after Definition 2.3.29. So, also a decision rule is chosen in such a way that the lower and upper bounds are met – provided that the upper and lower bounds do not constitute a contradiction.

The dual LP for SSO-MDPs with the action constraints equals:

$$\left| \begin{array}{l} \max \sum_{s \in \mathcal{W}} w(s) \\ \\ (J - P)^T x = \begin{cases} 1 & s = s_1 \\ -w(s) & s \in \mathcal{W} \\ -l(s) & s \in L \\ 0 & \text{else.} \end{cases} \\ \\ x_a \leq u(a) \cdot (J^T)_{s,*} x, \quad a = (s, \bar{a}), \quad a \in C \\ x_a \geq l(a) \cdot (J^T)_{s,*} x, \quad a = (s, \bar{a}), \quad a \in C \\ x \geq 0, \end{array} \right| \quad (\text{constraint SSO-MFP})$$

We want to show how this simple extension can be used to model the examples mentioned at the beginning of this subsection. Consider the sprinting soccer player. Assume the coach knows that sprinting in more than, e.g. 70% of playing situations will significantly impair the performance. We can model two kinds of sprinting actions with different outcomes, a recovered sprinting and an exhausted sprinting action. The recovered sprinting action should have a higher probability for a successful outcome than an exhausted sprint. In a standard SSO-MDP, the optimal decision rule would always select the recovered sprint. But introducing an upper bound of 0.7 on the recovered sprinting action forces the decision maker to decide between an exhausted sprint and a trot. Depending on the transitions probabilities of those two actions, he will decide to include trotting breaks in his strategy or not. Figure 3.10 sketches this example with transition probabilities such that the trot would be preferred over exhausted sprinting.

The critical point of this extension is the required expert knowledge. It may be difficult to determine the upper or lower bounds. Moreover, the estimation of the change in the transition probabilities when exceeding a bound needs specific observations.

3.8.2 Extension to Markov Games

Although it was argued at the beginning of this section that an MDP is more appropriate for modeling a sports game than a Markov game, it should be outlined how SSO-MDPs can be extended to sport-strategy-optimization Markov games (SSO-MGs).

In a Markov game, at each state, each team simultaneously selects its next action and the transition probabilities depend on the selected actions. Since a sports game consists of two teams, the parties participating in the game will be called in the following team 1 and team 2. The corresponding action sets in state s are denoted by \mathcal{A}_s^1 and \mathcal{A}_s^2 . The objective in an SSO-MG of team 1 is to maximize the expected total reward while the objective of team 2 is to minimize it. The reward functions of both teams are equal and summarized in one reward function. Like in SSO-MDPs there exists a starting state s_1 and goal states that are distinguished in winning states \mathcal{W} of team 1 and losing states L from the perspective of team 1.

Definition 3.8.2 (Sport-Strategy Optimization MG (SSO-MG)).

A *sport-strategy optimization Markov game (SSO-MG)* is an extended infinite-horizon MG (see Definition 2.5.1)

$$(S, I, \mathcal{A}_s^i, p(\cdot | s, a_1, a_2), r(s, a_1, a_2), \mathcal{W}, L, s_1)$$

with the following properties:

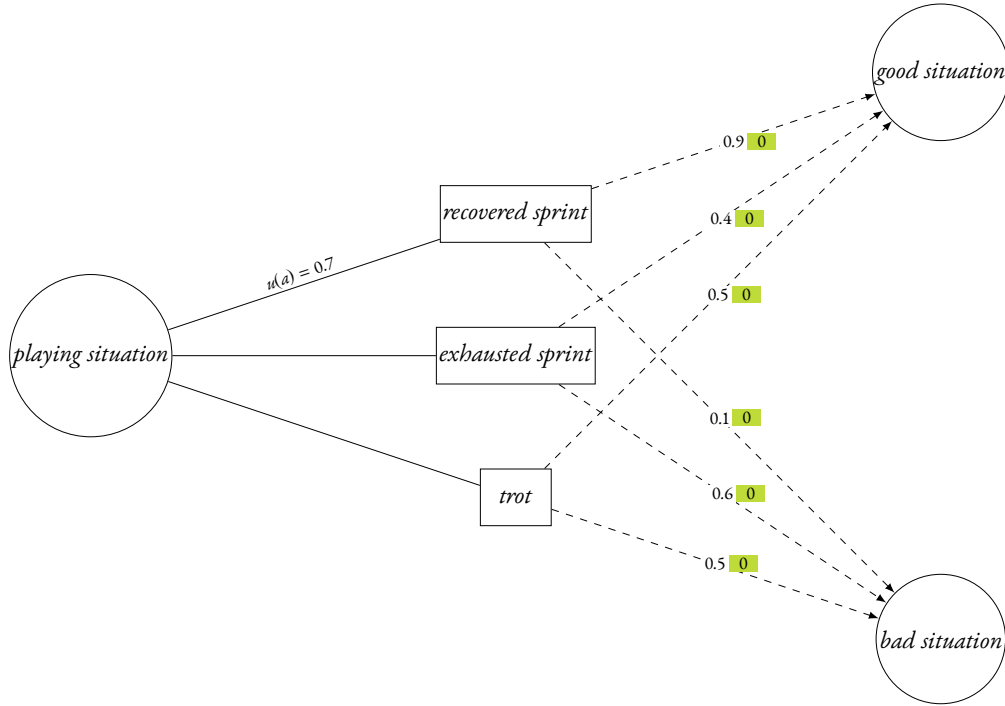


Figure 3.10: Modeling a sprint in soccer as constraint SSO-MDP

- S is a set of possible system states with a single-decision game defined for each state.
- I is the set of teams participating in the match with $I = \{1, 2\}$.
- A_s^i is the action set of team $i \in \{1, 2\}$ in state s .
- $p(\cdot | s, a_1, a_2)$ is a stationary transition probability function depending on the current state and the selected actions.
- $r(s, a_1, a_2)$ is the expected reward function of both teams (players). The reward function has the same properties as in an SSO-MDP:

$$r(s, a_1, a_2, s') = \begin{cases} 1, & \forall s \in S \setminus (W \cup L), s' \in W, a_1 \in A_s^1, a_2 \in A_s^2 \\ 0, & \text{else.} \end{cases}$$

- $W \subset S$ is the set winning states for team 1, which are losing states for team 2.
- $L \subset S$ is the set of losing states for team 1, which are winning states for team 2.
- $s_1 \in S$ is the known starting state.

Let π^i be a policy of team i and $\pi = (\pi^1, \pi^2)$ a policy configuration of both teams. The total expected reward from that policy configuration for both teams is

$$v^\pi(s) = \lim_{N \rightarrow \infty} \mathbb{E}_s^\pi \left\{ \sum_{t=1}^N r(X_t, Y_{1t}, Y_{2t}) \right\} = \lim_{N \rightarrow \infty} v_{N+1}^\pi(s).$$

Team 1 tries to maximize $v^\pi(s)$ while team 2 tries to minimize it. Since

$$\min_{\pi^2 \in \Pi_2} v^\pi(s) = - \min_{\pi^2 \in \Pi_2} -v^\pi(s),$$

the reward structure can be interpreted such that team 2 has to pay $v^\pi(s)$ to team 1. So, at each state s , a zero-sum matrix game is played where the payoffs are equal to the expected total reward:

$$\max_{\pi^1 \in \Pi_1} \min_{\pi^2 \in \Pi_2} v^{(\pi^1, \pi^2)}(s) = \min_{\pi^2 \in \Pi_2} \max_{\pi^1 \in \Pi_1} v^{(\pi^1, \pi^2)}(s).$$

This shows, that the SSO-MGs are two-player zero-sum Markov games as defined by Littman (1994).

Definition 3.8.3 characterizes Nash Equilibriums in SSO-MGs.

Definition 3.8.3 (Nash Equilibrium in SSO-MG).

A tuple of strategies (π^{1*}, π^{2*}) is a Nash equilibrium in an SSO-MG if

$$\pi^{1*} \in \arg \max_{\pi^1 \in \Pi_1} v^{(\pi^1, \pi^{2*})}(s)$$

and

$$\pi^{2*} \in \arg \min_{\pi^2 \in \Pi_2} v^{(\pi^{1*}, \pi^2)}(s).$$

holds for all $s \in S$.

The example SSO-MDP of Figure 3.1 has been extended to an SSO-MG by including actions for team 2 in each state. Furthermore, the example has been modified by introducing transitions from an action available in s_2 back to s_2 . The resulting SSO-MG example is presented in Figure 3.11.

The goal is to find a linear programming formulation for an SSO-MG by extending the dual LP for SSO-MDPs formulation of SSO-MDPs. As in MDPs, in MGs with stationary problem data, there always exists a stationary optimal policy. Consider as before for each team i a set of state-actions pairs A^i . A decision rule d^i used in a stationary policy of team i can be written as a vector with cardinality $m_i = |A^i|$. Let x_a , $a = (s, \tilde{a})$, $\tilde{a} \in A_s^1$ be the flow variable that corresponds to the decision rule d^1 of team 1 and y_b , $b = (s, \tilde{b}) \in A_s^2$ be the flow variable that corresponds to the decision rule of team 2. Then, the flow constraint of the dual LP for SSO-MDPs formulation for a state s that is not the starting state and no winning or losing state can be rewritten as

$$\sum_{a \in A_s^1, b \in A_s^2} x_a y_b - \sum_{a \in A^1, b \in A^2} p(s|a, b) x_a y_b = 0.$$

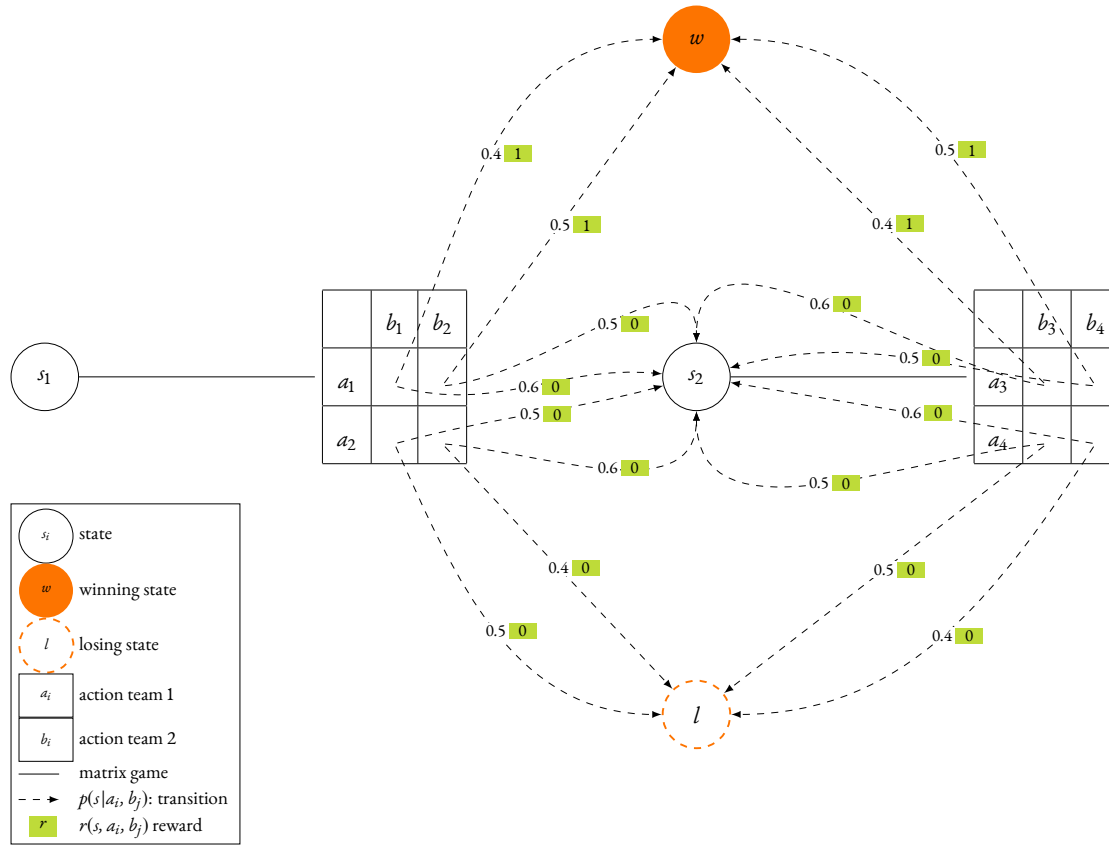


Figure 3.11: SSO-MG Example

The flow conditions in the starting state and in the absorbing states can be adapted analogously:

$$\begin{aligned}
 \sum_{a \in A_1^1, b \in A_2^2} x_a y_b - \sum_{a \in A^1, b \in A^2} p(s_1|a, b) x_a y_b &= 1 \\
 - \sum_{a \in A^1, b \in A^2} p(s|a, b) x_a y_b &= -w(s), \forall s \in W \\
 - \sum_{a \in A^1, b \in A^2} p(s|a, b) x_a y_b &= -l(s), \forall s \in L.
 \end{aligned}$$

Let m be the number of different action combinations of team 1 and team 2 occurring in the game. The maximum number of combinations is $m_1 \cdot m_2$. In the example of Figure 3.11, $m = 8$ and $m_1 \cdot m_2 = 16$. For technical considerations, an ordering of the combinations is defined by going rowwise through the matrices starting from the first action of team 1. The ordering of action combinations in the example of Figure 3.11 is

$$(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2), (a_3, b_3), (a_3, b_4), (a_4, b_3), (a_4, b_4).$$

Furthermore, let $C^1 \in \mathbb{R}^{m \times m_1}$ and $C^2 \in \mathbb{R}^{m \times m_2}$ be matrices that indicate whether an action is part of

the combination. The entry $C_{i,j}^k$ is 1 if action j of team k is part of the i -th action combination and 0 else. In the example of Figure 3.11, the combination matrices are

$$C^1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad C^2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Then by a pointwise multiplication of C^1x and C^2y , which is denoted by $C^1x \odot C^2y$, a vector with the flow value of an action combination can be determined.

Let P be in the context of SSO-MG be a transition matrix with m rows and n columns where each row corresponds to one action combination. The entry $P_{(a,b),i}$ equals the transition probability $p(i|a, b)$. Let $J \in \mathbb{R}^{m \times n}$ a matrix that indicates if action combination (a, b) is available in s . Then we can formulate a max-min problem for SSO-MGs:

$$\left| \begin{array}{l} \max_x \min_y \sum_{s \in W} w(s) \\ (J - P)^T (C^1x \odot C^2y) = \begin{cases} 1 & s = s_1 \\ -w(s) & s \in W \\ -l(s) & s \in L \\ 0 & \text{else.} \end{cases} \\ x, y \geq 0 \end{array} \right|$$

Obviously, this is not a linear program since there exit products of variables, and the objective function maximizes a minimum. Also, the relationship between a feasible solution of this optimization problem and a policy configuration needs to be examined. Probably an assumption like Assumption 3.2.1 for SSO-MDPs is necessary to draw conclusions from an optimal solution of this optimization problem.

However, this formulation is useful, if different opponents' strategies should be analyzed. For this purpose, the opponents' strategy can be set to a fixed probability distribution. The resulting maximization problem is an SSO-MDP if Assumption 3.2.1 is satisfied. Deriving an SSO-MDP from an SSO-MG has the advantage that the opponents' strategy is parametrized in the transition probabilities.

In the example of Figure 3.11, assume the opponent plays b_2 in s_1 and b_3 ins s_2 . Then an optimal strategy for team 1 would be to select in s_1 action a_1 and in s_2 action a_3 . With this strategy, team 1 wins with a probability of 1 against the opponent. So, in the case of this example, the SSO-MDP resulting from the SSO-MG does not satisfy Assumption 3.2.1.

If the value function of the SSO-MG is solved for every strategy combination, the result is a constant-sum matrix game with finite strategy sets of both teams. An example of this extension of an SSO-MDP to game theory can be found in Section 5.2.6.

Chapter 4

Application to Beach Volleyball

4.1 Introduction to Beach Volleyball

Beach volleyball was chosen as a first application of SSO-MDPs to a sports game. Beach volleyball belongs to the group of return plays, like tennis or badminton. In return plays, successful recovery of the ball to the opposing team or player is required. A net separates the court in two halves and on each half a team, or a player is located. Due to the return play and the division of the court in two halves, those sports games have a natural structure which is advantageous for modeling it as an MDP. Beach volleyball was chosen instead of tennis or badminton since in beach volleyball a team consists of two players. Therefore, there also exist some direct intra-team interactions. The modeling of player interactions may give some insights that are useful for sports games with larger team size, like handball or soccer, where many direct player interactions exist.

In this section, an overview of different modeling approaches for return plays is given. Afterward, the most relevant rules of beach volleyball are summarized.

4.1.1 Literature Overview – Modeling Return Plays

Some works on volleyball or beach volleyball using a Markov process approach have already been mentioned in Section 3.1 (Miskin, Fellingham, and Florence, 2010; Florence et al., 2008). In this subsection, some additional references to statistical investigations in connection with volleyball or beach volleyball are made, which do not necessarily use Markov chains. Furthermore, works from the field of informatics are presented that are related to beach volleyball and might have an impact on the application of SSO-MDPs.

Koch and Tilp found that the temporal position within a rally did neither affect the type nor the quality of the attack-hit (Koch and Tilp, 2009a). This result is an indication that stationary data can be reasonably assumed for a professional beach volleyball rally.

Buscà et al. investigated the influence of service characteristics on performance in men's and women's high-standard beach volleyball. When the speed of the ball was categorized into three groups, they found a relationship between serve ball speed and its effectiveness both for men and women (Buscà et al., 2012). These results were not observed when the speed was recorded using a radar gun. This investigation is interesting as the rally-SSO-MDP presented in Subsection 4.3.3 also requires a classification of whether

a ball was played hard or not.

In beach volleyball, a *call* is a suggestion of the setter to his or her partner where to place the attack in the opponent's court. Künzell et al. found that women use calls more often than men. Moreover for both the success rate of attacks increases with a call but the differences were only significant for women's beach volleyball (Künzell et al., 2014).

Natali et al. analyzed differences between males and females concerning the duration of point rallies, work rest ratio and the number of jumps and hits performed by the players according to their role (blockers and defenders). They did not show significant differences between males and females but between the jumps performed by blockers and defenders (Natali et al., 2017).

In 2002 the Fédération Internationale de Volleyball (FIVB) officially changed the scoring system from side-out scoring to rally point system. The goal was to stabilize the match duration. Several works are investigating the effects of the new scoring system like (Giatsis, 2003) and (Palao, Valades, and Ortega, 2012).

Cañal-Bruland, Mooren, and Savelsbergh showed that players and coaches, who have a perceptual-motor expertise, may contribute to successful action anticipation in beach volleyball (Cañal-Bruland, Mooren, and Savelsbergh, 2011). Successful action anticipation is not only useful for players or referees participating in a match. When a video analysis is done by hand like for generating the input data of the rally-SSO-MDP of Subsection 4.3.3 successful action anticipation is helpful.

There exist some works on tracking player's position and contact time points from videos in beach volleyball (Gomez et al., 2014). Advances in the field of automated tracking are of interest for the application of SSO-MDPs. A computerized tracking procedure that generates the required input data for SSO-MDPs is crucial such that SSO-MDPs with large numbers of parameters, like the rally-SSO-MDP defined in Subsection 4.3.3, can be set-up for a particular match in a short period. Cortell-Tormo et al. also track video sequences but for analyzing and comparing movement patterns and direction of locomotion in professional men's beach volleyball (Cortell-Tormo et al., 2011).

4.1.2 Summary of Beach Volleyball Rules

A short overview over beach volleyball based on the official beach volleyball rules 2017-2020 (Fédération Internationale De Volleyball, 2016) is given. Readers who are familiar with the rules of beach volleyball can skip this subsection without reservation. Rules regarding facilities and equipment, interruptions and delays, participants' conduct or referees will not be mentioned below and can be looked up in Fédération Internationale De Volleyball (2016).

Beach volleyball is a sports game between two teams playing on a court of sand, and each team is located on one half of the court. A team has three hits for returning the ball on the other court half. Beach volleyball is played according to a rally point system, which means that each win of a rally gives one point. The team who won the last rally gains the right to serve next. The serving player of each team must alternate each time the serving right is won.

A team wins a match when it has won two sets. A set is won if 21 points are scored by one team with a minimum lead of two points. In the case that each team has won one set, there is a deciding third set. The third set is played up to 15 points with a minimum lead of 2. If the minimum lead of 2 points is not fulfilled, the set goes on until one team has a lead of 2 points.

After this first, short characterization a more detailed description of some rules stated in Fédération Internationale De Volleyball (2016) is given. These rules are considered in the more detailed model of a

beach volleyball rally presented in Subsection 4.3.

Facilities:

The playing area includes the playing court and the free zone. The playing court is a rectangle measuring 16×8 meters, surrounded by a free zone, which is a minimum of 3 meters wide on all sides. The free playing space above the playing area shall measure a minimum of 7 meters in height. For FIVB, World and Official Competitions, the free zone must be between 5 and 6 meters, and free playing space must measure a minimum of 12.5 meters.

The service zone is located outside of the court behind the ground line of each team. It is 8 m wide and extends to the edges of the free space.

The net height is 2.43 m for men and 2.24 m for women. On each side of the net a 1.80 m long antenna is fastened.

Winning a Rally:

A rally is a sequence of playing actions. It starts with a serve and ends when the ball is out of play. If the rally results in the award of a point, it is called a completed rally. The team who wins the rally scores a point and serves next. If the team was the receiving team before, the serving player must be alternated. A team scores a point by successfully grounding the ball on the opponent's court, and when the opposing team commits a fault or receives a penalty.

Positions and the serve:

There are no determined positions on the court, except that each team must be within its court half at the moment the server hits the ball. The server himself is outside the court in the service zone and must not touch the court. When playing the ball, each team plays within its playing area. However, the ball may be retrieved from the free zone.

"In" and "Out":

A ball is "in" when it touches the surface of the playing court including the boundary lines. The ball is "out", when it falls on the ground completely outside the boundary lines, touches an object outside the court, crosses the vertical plane of the net either partially or totally outside the crossing space during a service or during the third hit of the team or crosses completely the lower space under the net. The allowed crossing space is the part of the vertical plane of the net limited by the top of the net from below and by the antennae and their imaginary extension at the side.

"Hits":

A "hit" is a contact with the ball by a player in the play. The ball may be touched with any part of the body. All actions which direct the ball towards the opponent, except for services and blocks, are considered as attack hits. Hits which preclude the ball from hitting the ground after the opponent team plays a serve are called receives. A *defending* is similar to a *receiving* with the difference that it is made to defend against an attack hit. By a *setting* the ball and the teammate are brought into a good position for the next *attacking hit*.

Faults:

When executing a hit, the ball must not be thrown or caught, else an *execution fault* occurs. There are more rules on how to execute a service, an attack hit or a block correctly. Infringement of these rules should be summarized under the term execution fault in this thesis.

A team has a maximum of three hits, including the block, for returning the ball over the net. If this maximum is exceeded a *four hits fault* occurs. A player must not hit the ball two times consecutively else a *double contact fault* happens. There is one exception of the double contact rule: The first hit after the block may be executed by any player including the one who has touched the ball during the block.

Two players may touch the ball simultaneously. Except for a simultaneous blocking this is counted as two hits.

Another fault is to complete an attack hit on the opponent's service when the ball is entirely higher than the top of the net. Furthermore, it is forbidden to block an opponent's service.

The basic structure of a rally is as follows: It starts with a serve of one team. The other team then receives the ball, sets the ball and attacks the serving team. As next, the serving team starts defending the attack hit and prepares its next attack by a set.

The sequence of receiving/defending – setting – attacking is called a complex. On average in professional men's beach volleyball matches there are two to three complexes during a rally. Ahmann found in his structure analysis Ahmann (2001) that 59% of the rallies in professional men's beach volleyball matches end after four or five ball contacts. Four or five ball contacts correspond to the sequence: serve – receive – setting – attack – block/defending. It should be pointed out that there need not be three ball contacts during a complex. Sometimes an attack hit at the second contact may be efficient if the opponent does not expect it. However, this occurs only in 6.3% of all played attacks in men's professional beach volleyball, see Ahmann (2001).

The summarized rules give an overview of possible situations that can occur in beach volleyball. Furthermore, they outline the available actions in a given situation of the game.

4.2 An SSO-MDP for a Beach Volleyball Set

In this section, an SSO-MDP is defined that models a beach volleyball set. The model should be designed as simple as possible while giving the opportunity to evaluate different serving and field attack strategies against a specified opponent team. The presented model is a generalization of the s-MDP defined in the working paper of Hoffmeister and Rambau (Hoffmeister and Rambau, 2017b). A first version that differs in larger parts from the one specified in this section was presented in Hoffmeister [formerly Börner] (2014).

4.2.1 Definition

Let team P be the team whose strategy should be optimized, and team Q be team P 's opponent. Assume team P can choose between finitely many serving strategies $serve_1, \dots, serve_{m_1}$ and finitely many field

attacks $attack_1, \dots, attack_{m_2}$. A field attack is considered as the whole sequence of

reception/defending – setting – attack hit.

It is assumed that team P can play any strategy at any time and wants to evaluate at which score in the set they should play which kind of serving or field attack strategy. Formally in an SSO-MDP, an optimal stationary decision rule is sought. In a sports context, it is common to speak of strategies instead of decision rules. Therefore, the term strategy will be used in this thesis if the context is less formal.

Based on team P 's question of interest, a state has to contain the current score, which team starts the next attack and an indicator of whether the state is a serving state or not. Thus, the simplest possible state space with respect to the regarded question is $\{(x, y, k, l) \mid x, y \in \mathbb{N}_0, k \in \{P, Q\}, l \in \{0, 1\}\}$. Here, x and y denote the scores of Team P and Q , respectively. The parameter k specifies which team possesses the ball, and l encodes whether it is a serving state ($l = 1$) or a field attack state ($l = 0$). If this state definition is used for the complete set including a tie, the result would be an infinite state space. Starting from a tie $(20, 20, k, l)$, there is an infinite number of states possible where no team gains a lead of 2 points.

As an SSO-MDP requires a finite number of states, a different state representation is used for the tie-game. Instead of remembering the number of points for team P and team Q separately, only the point difference of the two teams is denoted in a state. So, the states of the tie game are

$$S^{\text{tie}} = \{(z, k, l) \mid z \in \{-2, -1, 0, 1, 2\}, k \in \{P, Q\}, l \in \{0, 1\}\},$$

which are only finitely many states, namely 20. This kind of state representation is not possible in the regular set, since the absolute number of 21 points must be reached to win a set. In the tie-game, only a relative criterion must be fulfilled. The relative notation for the tie-game states affects that there are now finitely many states left that describe the regular game. These are

$$S^{\text{reg}} = \{(x, y, k, l) \mid x, y \in \{0, \dots, 21\} \text{ with } (x \leq 19 \vee y \leq 19), k \in \{P, Q\}, l \in \{0, 1\}\}.$$

The state set W should contain all states, where team P has won the set. The winning states of the regular game are all states where P has 21 points. Observe that the state set of the regular match contains only states where at least one team has no more than 19 points. So, all states of the regular state set where team P has 21 points are winning states. Furthermore, all states in the tie-game with $z = 2$ are winning states. Analogously, the state set L contains all states, where team P has lost the set, i.e., all states of the regular game where team Q has 21 points and all states of the tie-game with $z = -2$. As specified in the definition of an SSO-MDP, the states in $W \cup L$ are modeled as absorbing states.

A decision epoch starts when team P gains control over the ball and begins its field attack. The decision epoch ends when Team P makes a fault or a point, or when the offense is successful but Team Q gains control over the ball and starts its field attack. For each state $s = (x, y, k, l) \in S^{\text{reg}}$ or state $s = (z, k, l) \in S^{\text{tie}}$ that is a serving state of team P , i.e., a state with $k = P$ and $l = 1$, the action set in that state of team P is $A_s = \{serve_1, \dots, serve_{m_1}\}$. If the state is a field attack state of team P , which means $k = P$ and $l = 0$ in s , $A_s = \{attack_1, \dots, attack_{m_2}\}$. For all states with $k = Q$, the action set of team P is empty, i.e., $A_s = \emptyset$. In each absorbing state $W \cup L$, there exists an artificial action. However, those artificial actions are not explicitly listed in the following of this section.

As described in Subsection 3.8.2, it can be advantageous to model the opponent team analogously to the decision-making team. Therefore, the SSO-MDP is constructed with a symmetric view on the

teams P and Q . However, since an MDP and no MG is considered, team Q is assumed to play a fixed decision rule independent of the current score. So, there are no action sets defined for team Q and the fixed decision rule played by team Q is expressed by constant transition probabilities.

Let p_a [\bar{p}_a] be the probability that Team P playing action $a \in \mathcal{A}$, directly wins [loses] the rally. By directly, it is meant that there is no further field attack played by either team until the rally is completed. A superscript *serve* or *field* on p_a respectively \bar{p}_a denotes whether it was a transition belonging to a service or a field attack. The corresponding probabilities of Team Q are denoted by q and \bar{q} , respectively. As abbreviations, the probabilities that none of this happens is denoted by $\hat{p}_a := 1 - p_a - \bar{p}_a$ and $\hat{q} := 1 - q - \bar{q}$, respectively. So, with probability \hat{p}_a [\hat{q}] a subsequent field attack by the other team is started. In total, the evolution of the system is governed by the probabilities

$$p_a^{\text{attack type}}, \bar{p}_a^{\text{attack type}}, q^{\text{attack type}}, \bar{q}^{\text{attack type}},$$

where $a \in \mathcal{A}$, is the playing strategy and *attack type* $\in \{\text{serve}, \text{field}\}$ denotes the type of the attack.

These probabilities induce all transitions by incrementing points and changing the right to serve in the obvious way. All transitions that have a positive probability are explicitly listed in Table 4.1. Furthermore, in Figure 4.1, the resulting transition diagram for the case that P serves first in a simplified set is illustrated that requires only two (instead of 21) points for a win. From the states $(1, 0, k, l)$ and $(0, 1, k, l)$, $k \in \{P, Q\}$, $l \in \{0, 1\}$ a transition to one of the tie-game states $(0, P, 1)$ or $(0, Q, 1)$ is possible. These tie-game states correspond to states $(1, 1, P, 1)$ and $(1, 1, Q, 1)$ in the state notation of the regular game.

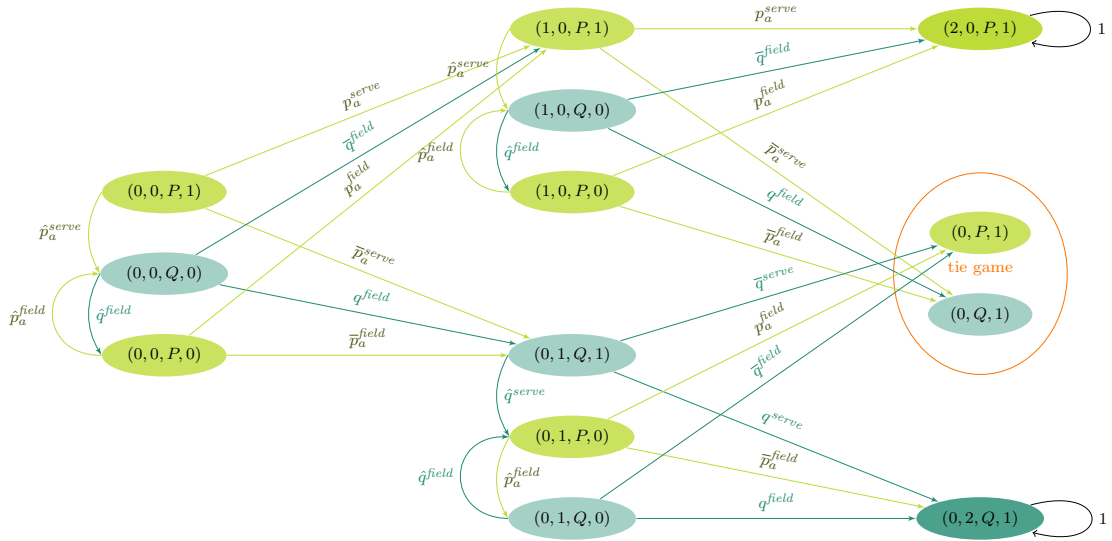


Figure 4.1: Set-SSO-MDP

The transitions of the tie-game are illustrated in Figure 4.2. They are divided into two sub figures. The left subfigure contains only transitions related to a service while the right subfigure contains all transitions that may occur after a field attack.

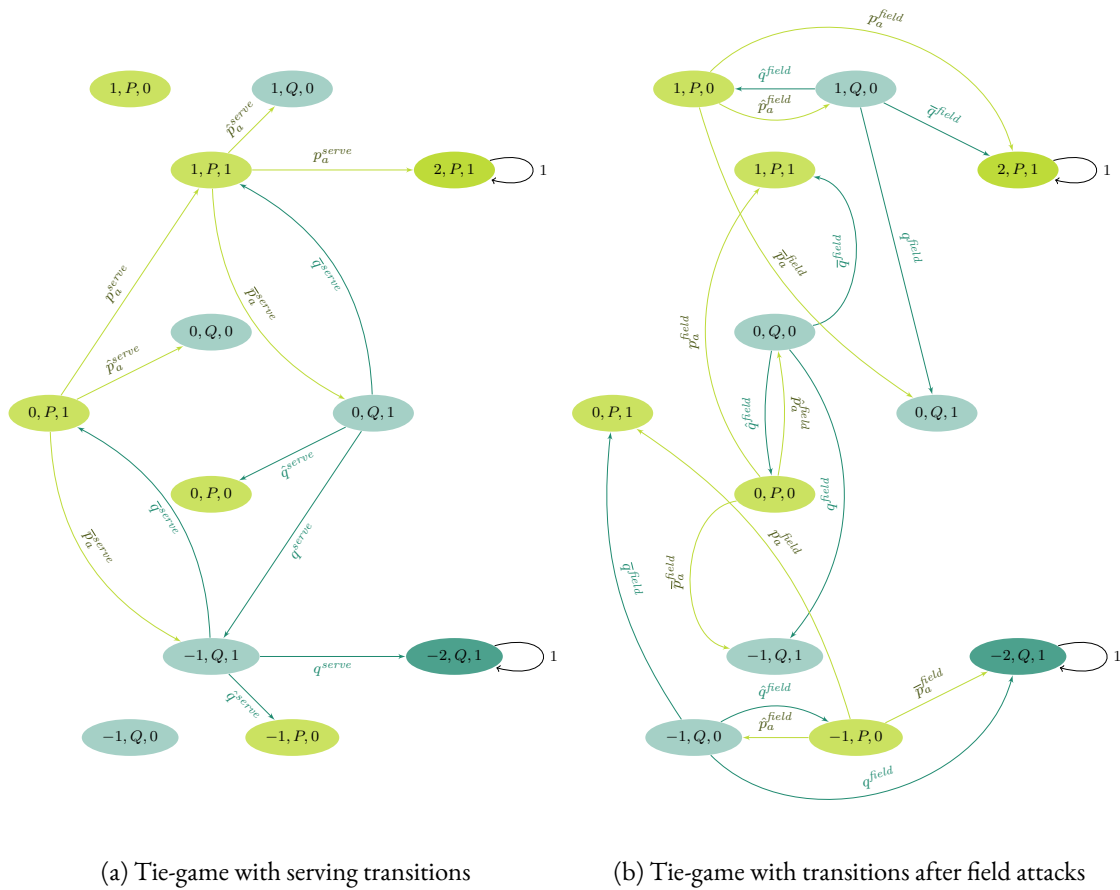


Figure 4.2: Tie-game

set-SSO-MDP	Beach Volleyball Set between Team P and Team Q
Decision Epochs:	$T = \{1, 2, 3, \dots\}$
State Sets:	$S = S^{\text{reg}} \cup S^{\text{tie}}$ $S^{\text{reg}} = \{(x, y, k, l) \mid x, y \in \{0, \dots, 21\} \text{ with } x \leq 19 \vee y \leq 19, \\ k \in \{P, Q\}, l \in \{0, 1\}\}$ $S^{\text{tie}} = \{(z, k, l) \mid z \in \{-2, \dots, 2\}, k \in \{P, Q\}, l \in \{0, 1\}\}$ $W = \{(21, y, k, l) \in S^{\text{reg}}\} \cup \{(2, k, l) \in S^{\text{tie}}\}$ $L = \{(x, 21, k, l) \in S^{\text{reg}}\} \cup \{(-2, k, l) \in S^{\text{tie}}\}$
Action Set:	$A_s = \begin{cases} \{serve_1, \dots, serve_{m_1}\} & \forall s = (x, y, P, 1) \in S^{\text{reg}}, s = (z, P, 1) \in S^{\text{tie}} \\ \{attack_1, \dots, attack_{m_2}\} & \forall s = (x, y, P, 0) \in S^{\text{reg}}, s = (z, P, 0) \in S^{\text{tie}} \\ \emptyset & \text{else.} \end{cases}$ <p>There exists an artificial action in each absorbing state $s \in W \cup L$.</p>
Transitions:	regular game and transition to tie-game
	<p>Let $s = (x, y, Q, 1) \in S^{\text{reg}} \setminus \{W \cup L\}$.</p> $p((x, y + 1, Q, 1) \mid s) = q^{\text{serve}} \text{ if } (x, y) \neq (20, 19), \quad p((0, Q, 1) \mid s) = q^{\text{serve}} \text{ if } (x, y) = (20, 19).$ $p((x + 1, y, P, 1) \mid s) = \bar{q}^{\text{serve}} \text{ if } (x, y) \neq (19, 20), \quad p((0, P, 1) \mid s) = \bar{q}^{\text{serve}} \text{ if } (x, y) = (19, 20).$ $p((x, y, P, 0) \mid s) = \hat{q}^{\text{serve}}$
	<p>Let $s = (x, y, P, 1) \in S^{\text{reg}} \setminus \{W \cup L\}, a \in A_s$.</p> $p((x + 1, y, P, 1) \mid s, a) = p_a^{\text{serve}} \text{ if } (x, y) \neq (19, 20), \quad p((0, P, 1) \mid s, a) = p_a^{\text{serve}} \text{ if } (x, y) = (19, 20).$ $p((x, y + 1, Q, 1) \mid s, a) = \bar{p}_a^{\text{serve}} \text{ if } (x, y) \neq (20, 19), \quad p((0, Q, 1) \mid s, a) = \bar{p}_a^{\text{serve}} \text{ if } (x, y) = (20, 19).$ $p((x, y, Q, 0) \mid s, a) = \hat{p}_a^{\text{serve}}$
	<p>Let $s = (x, y, Q, 0) \in S^{\text{reg}} \setminus \{W \cup L\}$.</p> $p((x, y + 1, Q, 1) \mid s) = q^{\text{field}} \text{ if } (x, y) \neq (20, 19), \quad p((0, Q, 1) \mid s) = q^{\text{field}} \text{ if } (x, y) = (20, 19).$ $p((x + 1, y, P, 1) \mid s) = \bar{q}^{\text{field}} \text{ if } (x, y) \neq (19, 20), \quad p((0, P, 1) \mid s) = \bar{q}^{\text{field}} \text{ if } (x, y) = (19, 20).$ $p((x, y, P, 0) \mid s) = \hat{q}^{\text{field}}$
	<p>Let $s = (x, y, P, 0) \in S^{\text{reg}} \setminus \{W \cup L\}, a \in A_s$.</p> $p((x + 1, y, P, 1) \mid s, a) = p_a^{\text{field}} \text{ if } (x, y) \neq (19, 20), \quad p((0, P, 1) \mid s, a) = p_a^{\text{field}} \text{ if } (x, y) = (19, 20).$ $p((x, y + 1, Q, 1) \mid s, a) = \bar{p}_a^{\text{field}} \text{ if } (x, y) \neq (20, 19), \quad p((0, Q, 1) \mid s, a) = \bar{p}_a^{\text{field}} \text{ if } (x, y) = (20, 19).$ $p((x, y, Q, 0) \mid s, a) = \hat{p}_a^{\text{field}}$

Transitions:	tie-game
Let $s = (z, Q, 1) \in S^{\text{tie}} \setminus \{W \cup L\}$.	Let $s = (z, P, 1) \in S^{\text{tie}} \setminus \{W \cup L\}$, $a \in A_s$.
$p((z-1, Q, 1) s) = \hat{q}^{\text{serve}}$,	$p((z+1, P, 1) s, a) = \hat{p}_a^{\text{serve}}$,
$p((z+1, P, 1) s) = \bar{q}^{\text{serve}}$,	$p((z-1, Q, 1) s, a) = \bar{p}_a^{\text{serve}}$,
$p((z, P, 0) s) = \hat{q}^{\text{serve}}$	$p((z, Q, 0) s, a) = \hat{p}_a^{\text{serve}}$
Let $s = (z, Q, 0) \in S^{\text{tie}} \setminus \{W \cup L\}$.	Let $s = (z, P, 0) \in S^{\text{tie}} \setminus \{W \cup L\}$, $a \in A_s$.
$p((z-1, Q, 1) s) = \hat{q}^{\text{field}}$,	$p((z+1, P, 1) s, a) = \hat{p}_a^{\text{field}}$,
$p((z+1, P, 1) s) = \bar{q}^{\text{field}}$,	$p((z-1, Q, 1) s, a) = \bar{p}_a^{\text{field}}$,
$p((z, P, 0) s) = \hat{q}^{\text{field}}$	$p((z, Q, 0) s, a) = \hat{p}_a^{\text{field}}$
$W \cup L$ are modeled as absorbing states and all other transitions have zero probability.	
Rewards:	$r(s, a, s') = \begin{cases} 1 & \text{if } s \notin W, s' \in W \\ 0 & \text{else.} \end{cases}$
Objective:	maximize the total expected reward

Table 4.1: An SSO-MDP modeling a beach volleyball set (set-SSO-MDP)

As in every SSO-MDP, entering a winning state yields a reward of one and all other transitions have a reward of zero. Table 4.1 summarizes the complete SSO-MDP for a beach volleyball set. It will be abbreviated in the following as set-SSO-MDP.

After having defined the set-SSO-MDP, we want to convince ourselves that Assumption 3.2.1 is satisfied. Since we consider playing strategies of a sports game, it is appropriate to assume that for no strategy $a \in A_s$ the transition probability \hat{p}_a is 1. If this would be the case, team P would always play that strategy, and no optimization problem would be needed. Also, it is appropriate to assume that each strategy has a positive probability to fail, which means, that \bar{p}_a is greater 0 for each $a \in A_s$. Of course, the same assumptions hold for the opponent's transition probabilities. From these properties, it can be concluded that under all strategies each state of the set-SSO-MDP is connected to a losing state by a path that has a probability greater than zero. So, Assumption 3.2.1 is satisfied and the presented MDP is an SSO-MDP according to Definition 3.2.2.

4.2.2 Transformation

The goal of this subsection is to redefine the set-SSO-MDP to get a representation that is better suited for mathematical analysis. The main idea is to remove cycles and to concatenate paths. In particular, states in which no decision is made by team P should be eliminated. The executed transformation in this subsection is in principle an application of the general transformation algorithm for SSO-MDPs presented in Subsection 3.7. However, the transformation algorithm is not followed step by step as the circles in the set-SSO-MDP are of a slightly different but simpler kind. Namely, as in a state where team Q is in possession of the ball the action set of team P is empty, there exist circles of the form

“state-state”. Of course, an artificial action could be inserted such that the transformation algorithm can be applied. However, this effort is saved, and the transformations carried out are explained more intuitively.

First the cycles between pairs of field attack states where only the team in possession of the ball differs should be eliminated, i.e., cycles between $(x, y, P, 0)$ and $(x, y, Q, 0)$ of the regular game respectively cycles between $(z, P, 0)$ and $(z, Q, 0)$ of the tie-game. The probability of staying in such a cycle is $\hat{p}_a \cdot \hat{q}$ which is by assumption of an SSO-MDP smaller than 1 since $\hat{p}_a = 1 - p_a - \bar{p}_a$ and \bar{p}_a is greater 0. The transformation will be illustrated on the example of a state belonging to the regular game. As the transition structure in the regular game is identical to that in the tie-game, this transformation can be analogously carried out for the states and transitions of the tie-game.

The state $(x, y, P, 0)$ should be kept and $(x, y, Q, 0)$ should be eliminated. The state $(x, y, Q, 0)$ has two outgoing transitions, one to $(x + 1, y, P, 1)$ and one to $(x, y + 1, Q, 1)$ in the regular game. At the states $(19, 20, Q, 0)$ the first transition is replaced by a transition to the state $(0, P, 1)$ of the tie-game and at the state $(20, 19, Q, 0)$ the second transition is replaced by a transition to the tie-game state $(0, Q, 1)$. For simpler notation, the transformation is illustrated at a state that has no transition to a tie-game state. Let $(x, y, P, 0)$ and $(x + 1, y, P, 1)$ be state nodes of the regular game. As there exists an optimal stationary and deterministic policy, the decision rule used in a state is time-independent. Furthermore, a deterministic decision rule chooses an action with certainty each time a state is met such that

$$p((x + 1, y, P, 1)|(x, y, P, 0), a) = p_a^{\text{field}} + \hat{p}_a^{\text{field}} \bar{q}^{\text{field}} + (\hat{p}_a^{\text{field}} \hat{q}^{\text{field}}) \cdot (p_a^{\text{field}} + \hat{p}_a^{\text{field}} \bar{q}^{\text{field}}) + (\hat{p}_a^{\text{field}} \hat{q}^{\text{field}})^2 \cdot \dots$$

As in the transformation algorithm, this is a geometric series. Its limit can be used to calculate the aggregated probability for a transition from $(x, y, P, 0)$ to $(x + 1, y, P, 1)$:

$$p((x + 1, y, P, 1)|(x, y, P, 0), a) = \frac{p_a^{\text{field}} + \hat{p}_a^{\text{field}} \bar{q}^{\text{field}}}{1 - \hat{p}_a^{\text{field}} \hat{q}^{\text{field}}}.$$

An analogous aggregation can be done for the transition form $(x, y, P, 0)$ to $(x, y + 1, Q, 1)$:

$$p((x, y + 1, Q, 1)|(x, y, P, 0), a) = \frac{\bar{p}_a^{\text{field}} + \hat{p}_a^{\text{field}} \hat{q}^{\text{field}}}{1 - \hat{p}_a^{\text{field}} \hat{q}^{\text{field}}}.$$

The eliminated circle has an incoming edge from the corresponding serving state $(x, y, k, 1)$. Since $(x, y, Q, 0)$ will be eliminated all transitions from $(x, y, k, 1)$ that go over $(x, y, Q, 0)$ to a subsequent state are replaced by a direct transition from $(x, y, k, 1)$. As an example,

$$p((x + 1, y, P, 1)|(x, y, P, 1), a) = p_a^{\text{serve}} + \hat{p}_a^{\text{serve}} \bar{q}^{\text{field}}.$$

Figure 4.3 illustrates the outcome of this transformation for the two-point set-SSO-MDP. For simpler notation of the result, the following terms are defined:

$$\alpha_a^{\text{field}, P} := \frac{p_a^{\text{field}} + \hat{p}_a^{\text{field}} \bar{q}^{\text{field}}}{1 - \hat{p}_a^{\text{field}} \hat{q}^{\text{field}}} \qquad \beta_a^{\text{field}, P} := \frac{\bar{p}_a^{\text{field}} + \hat{p}_a^{\text{field}} \hat{q}^{\text{field}}}{1 - \hat{p}_a^{\text{field}} \hat{q}^{\text{field}}}$$

The superscript *field* indicates that it is a transition from a field state and the subscript *P* indicates that the first field attack is executed by team *P*. The subscript *a* is the parameter for the field attack strategy. For each field attack strategy, the terms $\alpha_a^{field,P}$ and $\beta_a^{field,P}$ can be calculated. Note that $\alpha_a^{field,P} + \beta_a^{field,P} = 1$, which should be intuitive, since $\alpha_a^{field,P}$ is the probability for gaining the next point and $\beta_a^{field,P}$ for making the next fault at some point in the future. As one of these events must occur in a sports game, these probabilities should sum up to 1.

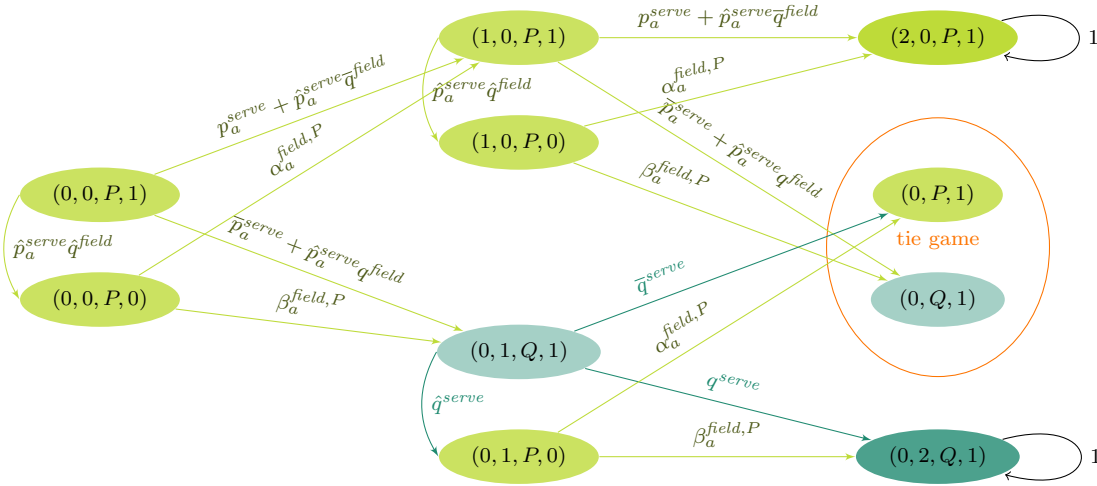


Figure 4.3: Refined regular SSO-MDP

By choosing not only the serving strategy but also the field attack strategy in a serving state of team *P*, the regular game can be further simplified. In a serving state of team *P*, the cumulated probability of gaining, or losing a point if serving strategy *a* and field attack strategy *b* is played, is:

$$\alpha_{a,b}^P := p_a^{serve} + \hat{p}_a^{serve} \cdot \bar{q}^{field} + \hat{p}_a^{serve} \cdot \hat{q}^{field} \cdot \alpha_b^{field,P}$$

$$\beta_{a,b}^P := \bar{p}_a^{serve} + \hat{p}_a^{serve} \cdot q^{field} + \hat{p}_a^{serve} \cdot \hat{q}^{field} \cdot \beta_b^{field,P}.$$

The terms are computed for every combination of a serving strategy *a* with a field attack strategy *b*. In a serving state of team *Q*, team *P* has only to choose a field attack strategy *b*. The cumulated probability of gaining, or losing, a point if field attack strategy *b* is played is:

$$\alpha_b^Q := \bar{q}^{serve} + \hat{q}^{serve} \cdot \alpha_b^{field,P} \quad \beta_b^Q := q^{serve} + \hat{q}^{serve} \cdot \beta_b^{field,P}.$$

Figure 4.4 shows the final transformed and aggregated regular game.

The tie-game can be transformed analogously. Figure 4.5 visualizes the tie-game using the introduced notation of the aggregated transition probabilities. Observe, that the cumulated probabilities for the next point of the aggregated tie-game are identical to the cumulated probabilities in the regular game.

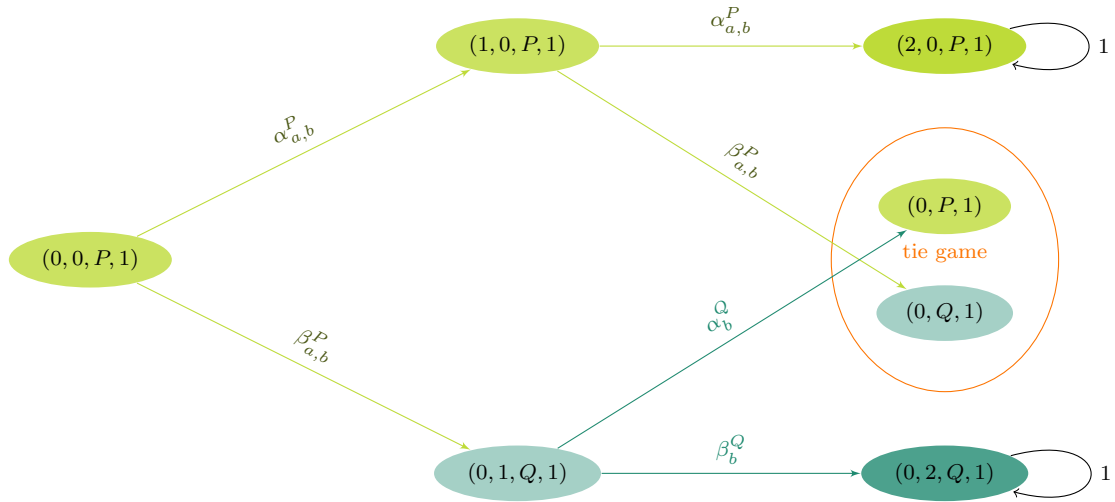


Figure 4.4: Aggregated regular SSO-MDP

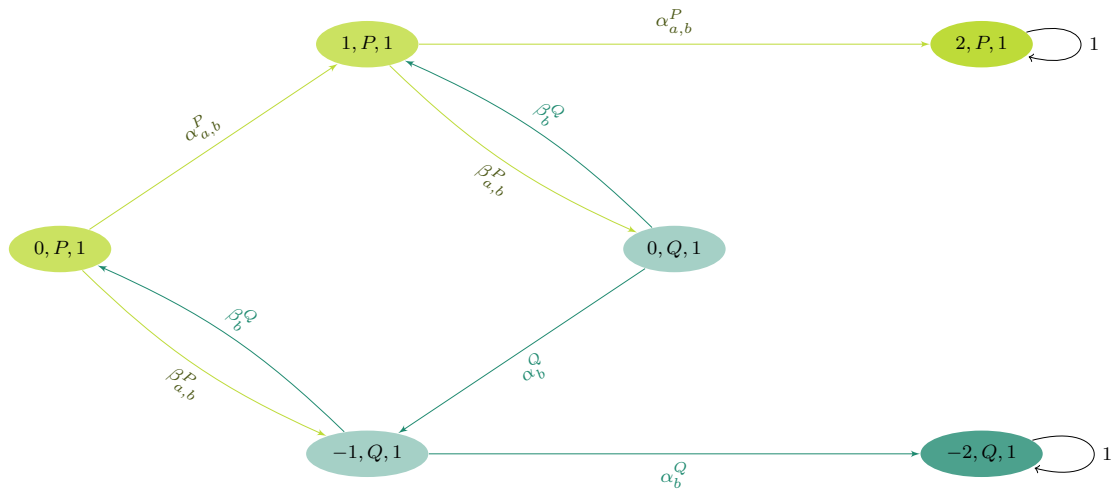


Figure 4.5: Aggregated Tie-Game

4.2.3 Mathematical Analysis

Some results of previous work that are related or can be applied to the presented set-SSO-MDP are:

An MDP for beach volleyball similar to the presented set-SSO-MDP was analyzed by Hoffmeister [formerly Börner] in (Hoffmeister [formerly Börner], 2014). The MDP presented in Hoffmeister [formerly Börner] (2014) does not contain a differentiation between serving states and field attack states. Hoffmeister [formerly Börner] showed a monotonicity property of the optimal value function. In Hoffmeister and Rambau (2017b), the authors present two MDPs for beach volleyball. The s-MDP of that manuscript has the same structure as the model presented here. The only difference is that the

s-MDP of Hoffmeister and Rambau (2017b) is already formulated for a certain benchmark question that considers a concrete set of playing strategies. Therefore, the results of Hoffmeister and Rambau are also applicable to the model presented here.

Two Lemmas are cited that are proven by Hoffmeister and Rambau in Hoffmeister and Rambau (2017b) and describe the monotonicity of the maximum total expected reward function. As proven in general, for SSO-MDPs the total expected reward function of a decision rule equals the probability of winning the match when playing according to the decision rule. The first lemma states that for a fixed team $k \in \{P, Q\}$ and a fixed type of state $l \in \{\text{serve}, \text{field}\}$, the maximum total expected reward is increasing in the own number of points and decreasing in the opponents number of points.

Lemma 4.2.1 (Hoffmeister and Rambau (2017b)):

The maximum total expected reward v^ of a state $(x, y, k, l) \in S$ satisfies*

$$v^*(x, y, k, l) \leq v^*(x + 1, y, k, l) \quad v^*(x, y, k, l) \geq v^*(x, y + 1, k, l),$$

for all $(x, y, k, l) \in S$.

The second lemma then describes a relation between serving states of different teams.

Lemma 4.2.2 (Hoffmeister and Rambau (2017b)):

The maximum total expected reward v^ in a serving state satisfies*

$$v^*(x + 1, y, P, 1) \geq v^*(x, y + 1, Q, 1),$$

for all $(x, y, k, l) \in S$.

The proofs of both lemmas can be found in Hoffmeister and Rambau (2017b). They can also be applied to the tie-game, as the structure of transitions is identical.

Due to the described monotonicity property, a myopic policy that maximizes the probability to win the next point is optimal. This result was independently developed from Walker, Wooders, and Amir (2011), who proved that given a monotonicity property a myopic policy is optimal for binary Markov games.

Since the transition probabilities are identical in every stage of the game, the optimal myopic policy stays the same throughout the game. Due to the structure of the transition probabilities, it is enough to maximize the point probability for the best service-field attack combination in a service state of team P . The determined field attack strategy is then also optimal for rallies where team Q is the serving team since $\alpha_{a,b}^P$ is increasing in the point probability of a field attack state. The main theoretical result for the analytic solution of the SSO-MDP is the following:

Theorem 4.2.3 (Optimal Policy):

There exists a stationary optimal policy that chooses in each serving state $(x, y, P, 1) \in S^{\text{reg}}$ [$(z, P, 1) \in S^{\text{tie}}$] the serving strategy $a^ \in \{\text{serve}_1, \dots, \text{serve}_{m_1}\}$ and in each non serving state $(x, y, P, 0) \in S^{\text{reg}}$ [$(z, P, 0) \in S^{\text{tie}}$] the field attack strategy $b^* \in \{\text{attack}_1, \dots, \text{attack}_{m_2}\}$ with*

$$\alpha_{a^*, b^*}^P \geq \alpha_{a, b}^P, \forall a \in \{\text{serve}_1, \dots, \text{serve}_{m_1}\}, b \in \{\text{attack}_1, \dots, \text{attack}_{m_2}\}.$$

PROOF. After the transformation, the optimality equations in a serving state $(x, y, P, 1)$ are:

$$v^*(x, y, P, 1) = \max_{\substack{a \in \text{serve}_1, \dots, \text{serve}_{m_1}, \\ b \in \text{attack}_1, \dots, \text{attack}_{m_2}}} \left\{ \alpha_{a,b}^P v^*(x+1, y, P, 1) + (1 - \alpha_{a,b}^P) v^*(x, y+1, Q, 1) \right\}$$

It can be directly followed that team P chooses the combination a, b that maximizes $\alpha_{a,b}^P$ since $v^*(x+1, y, P, 1) \geq v^*(x, y+1, Q, 1)$ by Lemma 4.2.2.

For a non-serving state $(x, y, P, 0) \in S^{\text{reg}}$ [$(z, P, 0) \in S^{\text{tie}}$], the cumulated probability for reaching $(x+1, y, P, 1)$ [$(z+1, P, 1)$] is $\alpha_b^{\text{field}, P}$ and for reaching $(x, y+1, Q, 1)$ [$(z-1, Q, 1)$] is $\beta_b^{\text{field}, P}$, compare Figure 4.3. Applying Lemma 4.2.2 again, it can be derived that team P tries to maximize $\alpha_b^{\text{field}, P}$. As

$$\alpha_{a,b}^P = \hat{p}_a^{\text{serve}} + \hat{p}_a^{\text{serve}} \cdot \hat{q}^{\text{field}} + \hat{p}_a^{\text{serve}} \cdot \hat{q}^{\text{field}} \cdot \alpha_b^{\text{field}, P},$$

$\alpha_{a,b}^P$ is monotonically increasing in $\alpha_b^{\text{field}, P}$. Therefore, if a^*, b^* is a maximizing combination of $\alpha_{a,b}^P$ and $\hat{p}_a^{\text{serve}} \cdot \hat{q}^{\text{field}} < 0$ holds, it follows that b^* is also a maximizer of $\alpha_b^{\text{field}, P}$. ■

4.2.4 Winning Probability of the Tie-Game

It is even possible to find an analytical expression for the probability of winning the tie-game. In the following the winning probability $v(z, P, 1)$ of team P under the serving strategy a and the field attack strategy b is abbreviated by $v_{a,b}^{z,P}$. Analogously, the winning probability $v(z, Q, 1)$ of team P under the playing strategy a, b is denoted by $v_{a,b}^{z,Q}$. Due to the monotonicity lemmas and the structure of the optimal policy, compare Theorem 4.2.3, there exists a unique combination a, b of strategies which is played in each state of the tie-game. Therefore, the following system of equations holds for $v_{a,b}^{0,P}$ and $v_{a,b}^{0,Q}$:

$$\begin{aligned} v_{a,b}^{0,P} &= \alpha_{a,b}^P \cdot v_{a,b}^{1,P} + \beta_{a,b}^P \cdot v_{a,b}^{-1,Q} \\ v_{a,b}^{1,P} &= \alpha_{a,b}^P \cdot 1 + \beta_{a,b}^P \cdot v_{a,b}^{0,Q} \\ v_{a,b}^{0,Q} &= \alpha_b^Q \cdot v_{a,b}^{1,P} + \beta_b^Q \cdot v_{a,b}^{-1,Q} \\ v_{a,b}^{-1,Q} &= \alpha_b^Q \cdot v_{a,b}^{0,P} + \beta_b^Q \cdot 0 \end{aligned}$$

Solving this system of equations yields to

$$v_{a,b}^P := v_{a,b}^{0,P} = \frac{(\alpha_{a,b}^P)^2}{(1 - \alpha_b^Q \beta_{a,b}^P)^2 - \alpha_{a,b}^P \alpha_b^Q \beta_{a,b}^P \beta_b^Q} \quad v_{a,b}^Q := v_{a,b}^{0,Q} = \frac{\alpha_{a,b}^P \alpha_b^Q (\alpha_{a,b}^P \beta_b^Q - \alpha_b^Q \beta_{a,b}^P + 1)}{(1 - \alpha_b^Q \beta_{a,b}^P)^2 - \alpha_{a,b}^P \alpha_b^Q \beta_{a,b}^P \beta_b^Q}. \quad (4.1)$$

After this analysis, different service and field attack strategies can be compared if the governing transition probabilities are known.

4.2.5 Application to a Match

The beach volleyball final of Olympic games 2012 in London was chosen as a first application of the set-SSO-MDP. In the final match, the German team Brink-Reckermann won against the Brazilians Alison-Emanuel in three sets (23 : 21, 16 : 21, 16 : 14). The final match and all pre-final matches of the finalists are publicly available on the Olympics YouTube channel¹.

From the video recording of the final match, it is possible to estimate the transition probabilities of the set-SSO-MDP for the played decision rules in the final match. For this purpose, every played rally in the final match was recorded as a sequence of a service and subsequent field attacks of the participating teams. This annotation was carried out with the support of a student, Fabian Buck, who has practical experiences in beach volleyball. The services were denoted by *Serve P* respectively *Serve Q* depending on whether it was a serve of team *P* or a serve of team *Q*. The field attacks are abbreviated by *Field P* and *Field Q*. With this notation, e.g., the first rally of the final match is

$$\textit{Serve Q} - \textit{Field P} - \textit{Point Q},$$

where team *Q* are the Brazilians and team *P* the Germans. The collected rallies were saved together with a time-stamp of their position in the video in a text file with the extension `.sdata`. The extension is called `.sdata` because in Chapter 5.2 the set-SSO-MDP will be called s-MDP. From the annotated rallies, the set-SSO-MDP transition probabilities of the played decision rules can be estimated by a maximum likelihood estimation as

$$\begin{aligned} p^{serve} &= \frac{\#\textit{“Serve P - Point P”}}{\#\textit{“Serve P”}} & p^{field} &= \frac{\#\textit{“Field P - Point P”}}{\#\textit{“Field P”}} \\ \bar{p}^{serve} &= \frac{\#\textit{“Serve P - Point Q”}}{\#\textit{“Serve P”}} & \bar{p}^{field} &= \frac{\#\textit{“Field P - Point Q”}}{\#\textit{“Field P”}} \\ \hat{p}^{serve} &= \frac{\#\textit{“Serve P - Field Q”}}{\#\textit{“Serve P”}} & \hat{p}^{field} &= \frac{\#\textit{“Field P - Field Q”}}{\#\textit{“Field P”}}. \end{aligned}$$

The number of the matching substrings in the s-data strings can, e.g., be determined by a search with regular expressions. A technical note: There exists a small number of rallies in the s-data that contain the substring *Field P-Field P* or *Serve P-Field P*. Such sequences were recorded when a ball was blocked and went back over the net or when the opposing team did not play a full field attack, but the ball crosses the net for instance at the reception. In this count, we evaluated *Field P-Field P* and *Serve P-Field P* as a case where a subsequent field attack and no direct point or fault follows. So, the number of occurrences of these sequences were added to the numerator of \hat{p}^{serve} respectively \hat{p}^{field} . The corresponding transition probabilities of the opponent team can be estimated analogously. For the final match and the final strategy, abbreviated by *final*, we estimated the following transition probabilities for the German team Brink-Reckermann:

$$\begin{aligned} p_{final}^{serve} &= \frac{1}{55} \approx 1.8\% & p_{final}^{field} &= \frac{34}{70} \approx 48.6\% \\ \bar{p}_{final}^{serve} &= \frac{2}{55} \approx 3.6\% & \bar{p}_{final}^{field} &= \frac{12}{70} \approx 17.1\% \\ \hat{p}_{final}^{serve} &= \frac{52}{55} \approx 94.5\% & \hat{p}_{final}^{field} &= \frac{24}{70} \approx 34.3\% \end{aligned}$$

¹<https://www.youtube.com/user/olympic>

For the Brazilian team Alison-Emanuel, we estimated the transition probabilities:

$$\begin{aligned} q_{final}^{serve} &= \frac{2}{56} \approx 3.6\% & q_{final}^{field} &= \frac{40}{73} \approx 54.8\% \\ \bar{q}_{final}^{serve} &= \frac{8}{56} \approx 14.3\% & \bar{q}_{final}^{field} &= \frac{12}{73} \approx 16.4\% \\ \hat{q}_{final}^{serve} &= \frac{46}{56} \approx 82.1\% & \hat{q}_{final}^{field} &= \frac{21}{73} \approx 28.8\% \end{aligned}$$

The counted number of direct points and errors after a service or a field attack are presented in the numerator of the fractions and can be compared to publicly available match statistics, for instance, Database (2018). The match statistics provided by Database (2018) contain the number of aces, service errors, kills and attack errors. A kill is a field attack that can not be defended and leads to a point. Table 4.2 shows the match statistic after Database (2018). The number of aces equals the number of direct points we counted after a service and also the number of service errors equals our counted number of faults in a serving situation. The total number of services is not available in this source, and we could not find any other source related to the Olympic final of 2012 containing the total number of services. The total number of attacks, observed by us, is larger than the total number of field attacks listed in the match statistic. A reason for this may be that we also counted a reception or defense followed by a set and a planned shot as a field attack. It may be the case that only smashes were counted in the match statistics. The number of kills equals our counted number of direct points after field attack for the Germans. However, we counted one more direct point after a field attack for Brazil. The most significant difference between our counted values and the presented statistic is the number of errors. The match statistic states that no field attack errors occurred. However, an example of an attack fault which proves that zero attack errors cannot be correct is: In the very first rally of the final match, Brazil starts with a serve and the subsequent field attack of Germany is blocked such that a point for Brazil is counted. Another example is the field attack of Germany at 10 minutes and 2 seconds, where the attack hit of Brink goes behind the baseline and the ball is out. We classify both examples as faults after a field attack hit. The reader may watch this sequence in the video <https://www.youtube.com/watch?v=H7iQ4sAf00E> on YouTube.²

Player	attacks	kills	errors	services	aces	service errors
Brink	25	12	0	?	0	1
Reckermann	39	22	0	?	1	1
Alison	17	12	0	?	1	5
Emanuel	57	29	0	?	1	3

Table 4.2: Match statistics of the final match at the Olympic Beach Volleyball Tournament 2012 in London (Database, 2018)

It may be interesting to determine how large the winning probability of Germany is in our model given the estimated transition probabilities for the played strategy in the final match. Formula 4.1 can

²Through an e-mail conversation with the administrator of the www.bvbinfo.com website (that hosts the beach volleyball database), I got the information that the numbers of attack errors were not available for that tournament.

be used to calculate the winning probability of Germany in the tie-game which results in

$$\begin{aligned} v_{a,b}^P &= \frac{(\alpha_{a,b}^P)^2}{(1 - \alpha_b^Q \beta_{a,b}^P)^2 - \alpha_{a,b}^P \alpha_b^Q \beta_{a,b}^P \beta_b^Q} \\ &\approx \frac{(0.3372)^2}{(1 - 0.6369 \cdot 0.6628)^2 - 0.3372 \cdot 0.6369 \cdot 0.6628 \cdot 0.3631} = 0.4029, \end{aligned}$$

The variables $\alpha_{a,b}^P$, $\beta_{a,b}^P$, α_b^Q and β_b^Q are computed according to the formulas of Subsection 4.2.3. This result means that if Germany starts with a serve at a score of 19 : 19, the probability that Germany achieves a lead of 2 points and wins the tie-game is 40.29%. The value $v_{a,b}^P$ can be used as a terminal reward in state (19, 19, P, 1) of the regular game, and $v_{a,b}^Q$, which equals 0.5328, as a terminal reward in state (19, 19, Q, 1). With this modification the regular game is a finite MDP and can be solved by dynamic programming.³ By dynamic programming, it follows that Germany has a probability of 40.8% for winning a set if the set starts with a serve of Germany and a winning probability of 44.66% if the set starts with a serve of Brazil. Following these computational results of the set-SSO-MDP, one would suggest that it has been more likely that Brazil wins the Olympic final. However, Germany has won the match. So, the question may arise whether these findings tell something about the model-validity? A major challenge of sport-strategy optimization is that each match or set is just one realization of the random process. It may be the case, that the model maps the dynamics of the match well, and in the long run Germany will only win around 43% of the sets, but in the small sample of 3 sets, it may nevertheless be that the Germans win 2 of 3 sets and become Olympic Champions. However, it may also be the case, that the model does not reflect reality well enough.

However, the main purpose of sport-strategy optimization is to give strategic recommendations, and the absolute winning probability is not the most important information. It may be more valuable information to know which strategy from a given set of strategies has the highest winning probability and is relatively the best strategy. Given a set of strategies together with the transition probabilities p_a , \bar{p}_a and \hat{p}_a for each strategy a in the serving and the field attack situation, it is easy to evaluate which strategy is the best. One only needs to calculate the winning probability of each strategy, like it was done for the final strategy, and compare them to each other. The strategy with the highest winning probability is the best strategy compared to the other strategies. Alternatively, even easier, it is enough to calculate the aggregated probability for the next point of each strategy, which is $\alpha_{a,b}^P$ and compare those values to each other. According to Theorem 4.2.3, the strategy with the largest $\alpha_{a,b}^P$ is the strategy with the highest winning probability.

The difficulty of giving a strategic recommendation using this set-SSO-MDP does not lie in the evaluation of a strategy. However, it is much more difficult to estimate the correct transition probabilities of a particular strategy. The transition probabilities in the set-SSO-MDP depend on both teams, since, e.g., at a field attack, the quality of the hit of the attacking team, and the defending skills of the opponent team affect whether the attack is a direct point, a fault or is successfully defended. Due to the dependence on the opponent team, the transition probabilities should be estimated only from rallies of matches between the teams under investigation. However, the number of matches between the same teams

³Formally, also the artificial actions in the absorbing state have to be removed to make the SSO-MDP a finite-horizon MDP. The absorbing states L and W have to be changed to terminal states with a terminal reward of 0.

during a season or a particular tournament is not high. For example at the Olympic games 2012 in London, Brink-Reckermann did not face Alison-Emanuel before the final match.

Nevertheless, assume it should be compared whether it is better for Germany to play at high risk or to play safe. Let *risky* be a strategy that is characterized by jump serves into border fields in the serving situation and smashes into border fields in field attack situations. On the other hand, the strategy *safe* should be characterized by float serves and planned shots both towards non-border fields.

The final is used to estimate the transition probabilities of the different serving and field attack strategies. After classifying all serves and field attacks of the final, 1 risky serve of Germany and 38 safe serves are counted. Compared to the total number of 55 serves of Germany in the final match, 16 serves could not be classified as *risky* or *safe*. Serves cannot be classified as *risky* or *safe* if a jump-serve is played in a non-border field or a float serve into a border field. In the field attack situation, only 23 of 70 field attacks could be classified. A field attack can not be classified if a smash is played in a non-border field or a planned shot into a border field. Table 4.3 summarizes the estimation results of the transition probabilities for the strategies *risky* and *safe* based on the final match. The number of observations got smaller for estimating transition probabilities according to defined strategies. If the number of strategies increases or the strategies are characterized by more specific criteria, the number of observations will become even smaller.

Transition probabilities based on the final match						
strategy	#	p^{serve}	\bar{p}^{serve}	#	p^{field}	\bar{p}^{field}
<i>risky-risky</i>	1	0%	0%	12	42%	25%
<i>risky-safe</i>	1	0%	0%	11	64%	9%
<i>safe-risky</i>	38	3%	0%	12	42%	25%
<i>safe-safe</i>	38	3%	0%	11	64%	9%

Table 4.3: Estimated transition probabilities from the final match

Nevertheless, given the estimated transition probabilities for *risky* and *safe* in the serving and the field attack situation, Theorem 4.2.3 can be applied to compute an optimal decision rule. The strategy *safe-safe* generates the largest aggregated probability of winning the next point, which is $\alpha_{safe, safe}^P$. Therefore, according to Theorem 4.2.3, it is optimal to play throughout the whole set floats serves and planned shots into non-border fields, which is the characterization of *safe-safe*. If the winning probability of each strategy combination is computed by dynamic programming, the resulting values confirm our findings of Theorem 4.2.3. All estimated transition probabilities and winning probabilities are summarized in Table 4.4.

It is remarkable that the winning probability of the optimal strategy *safe-safe* is markedly greater than 50%. The actual played final strategy *final* has only a winning probability of 43%. Both strategies are evaluated against the same opponent. So, the model suggests that choosing the right strategy can make the difference on who wins the match more likely. It is impossible to test whether Germany would have won the final match more clearly if they had played *safe-safe*. Moreover, even if it would be possible to reproduce the Olympic final and let Germany play *safe-safe* against the Brazilians in London, several sets are needed to get a proper sample size. In any case, the findings should always be combined with

Based on rally records of final match								
strategy $a-b$	#	p_a^{serve}	\bar{p}_a^{serve}	#	p_b^{field}	\bar{p}_b^{field}	$\alpha_{a,b}^P$	winning Prob
<i>risky-risky</i>	1	0%	0%	12	42%	25%	31%	21%
<i>risky-safe</i>	1	0%	0%	11	64%	9%	38%	82%
<i>safe-risky</i>	38	3%	0%	12	42%	25%	33%	25%
<i>safe-safe</i>	38	3%	0%	11	64%	9%	39%	85%
<i>final</i>	55	2%	4%	70	49%	17%	34%	43%
	#	q^{serve}	\bar{q}^{serve}	#	q^{field}	\bar{q}^{field}		
<i>final</i>	56	4%	14%	73	55%	16%		

Table 4.4: Comparing strategies against the final strategy of Brazil

expert knowledge to confirm the results or to reject them as an artifact.

The question may arise whether it is possible to give such a strategic recommendation prior to the final match. As mentioned above, due to the dependence of the transition probabilities on both teams, matches between Brink-Reckermann and Alison-Emanuel a priori the final would have been needed. A way out could be to classify other teams participating at the Olympic games according to their similarity to the Brazilian team. With such a proceeding it could be possible to use the pre-final matches and get a more significant number of observations.

The general problem of this rough set-SSO-MDP is to find data that suites to the analyzed strategies. The more is captured by the transition probabilities, the more properties must be fulfilled such that a match can be used as an information base to estimate transition probabilities. However, the advantage of this rough SSO-MDP is that it is easy to determine an optimal decision rule or to compute the winning probability of a particular decision rule.

4.3 An SSO-MDP for a Beach Volleyball Rally

This section defines a second infinite-horizon, stationary SSO-MDP for beach volleyball. This time the model captures only a beach volleyball rally instead of a complete set. The presented model in this section is identical to the g-MDP presented in the manuscript Hoffmeister and Rambau (2017b). However, the system dynamic and strategy definition are presented in more detail. The motivation for the very technical description is that the rally-SSO-MDP could be reimplemented by the reader.

Let again P and Q be the teams participating in the match. P_1 and P_2 should be the players of team P ; Q_1 and Q_2 the players of team Q . As before, team P is the team whose playing strategy shall be optimized, whereas team Q is the uncontrolled opposing team. As in the set-SSO-MDP, team P is the decision-making team, and the behavior of team Q is part of the system disturbance and included in the transition probabilities. However, the model is again built with a symmetric view on team P and team Q . So, team Q 's action sets will be analogously modeled to the action sets of team P while team Q plays a fixed probability distribution over the available actions. The transition probabilities are determined by the randomized choices of Q 's actions and the system disturbances.

As seen in the last section, if large parts of the game dynamics are included in one transition, it gets hard to find suitable data for estimating the transition probabilities. This problem can be avoided by modeling the game mechanism in a rally more explicitly. As a consequence, the transitions decompose and capture smaller parts of the game mechanism. So, it gets easier to find appropriate data that fits the considered transition. The focus of the rally-SSO-MDP should be on estimable transition probabilities while accepting a probably large and complex SSO-MDP. A team action will be split up into two individual player actions. Moreover, any player action will be defined as a combination of a hit and a move. The decomposition of a team action into actions of single players will allow building the SSO-MDP solely on individual player probabilities.

4.3.1 Definition

The decision epochs are modeled as the points in time at which one of the players hits the ball, or the ball touches the ground. If some player contacts the ball during a blocking action or accidentally with some part of his body, no decision point should be defined. Each time a decision point occurs, the current state is observed, and each player has to decide about his next hit and his next movement. The rally is completed when the ball hits the ground, or a player makes a fault. As it is characteristic for SSO-MDPs, the rally will end for sure after a finite but undefined number of contacts. The points in time in a rally, where the state of the system is observed, define the decision epochs $T = \{1, 2, 3, \dots\}$ where $t \in T$ is the total number of ball contacts minus the blocking contacts in the rally so far.

The court is divided in a grid as the position of a player relative to the ball is important to determine which player can perform a hit, and as the absolute position on the court influences the player's hitting performance. The grid is presented in Figure 4.6. The exact division of the court was determined by evaluating the observed data of hits which were used to estimate the individual player skills. As described in Section 4.3.3, the original data was saved with exact coordinates and made it possible to examine different divisions of the field. The presented division is the one that was finally used. However, when considering a match on a different skill level or concerning a different issue, a modification of the grid is generally possible.

In the following, a function $pos(\cdot)$ is used that returns the position of a player or the ball according to the specified grid. A state is defined as a combination of several state variables. The complete state space is *factored* into constituent variables which are the players' positions, the ball's position, a counter for the number of contacts, the information which player last contacted the ball, a boolean variable that indicates the hardness of the last hit and the designated blocking player of the defending team for the next attack. A general formulation of a state is

$$(pos(P_1), pos(P_2), pos(Q_1), pos(Q_2), pos(ball), counter, lastContact, hardness, blockingPlayer).$$

Such a description of an MDP is called a *factored MDP* (Kolobov and Mausam, 2012, Def. 2.22, p. 23). The constituent variables are also called *feature* variables. The domains of the feature variables are

$$\begin{aligned} dom(pos(P_1)) &= dom(pos(P_2)) = \{P00, P01, \dots, P34, P35\} \\ dom(pos(Q_1)) &= dom(pos(Q_2)) = \{Q00, Q01, \dots, Q34, Q35\} \\ dom(pos(ball)) &= \{P00, P01, \dots, P34, P35, Q00, Q01, \dots, Q34, Q35\} \\ dom(counter) &= \{-1, 0, 1, 2, 3\} \end{aligned}$$

category	description	counter	further requirement
S^{rec}	receiving state	-1	not in $S_P^{serve} \cup S_Q^{serve}$
S^{def}	defending state	0	
S^{set}	setting state	1	
S^{attack}	attacking state	2	

Table 4.5: State Categories

$$(P12, P03, Q12, Q13, P03, -1, \emptyset, 0, \emptyset)\},$$

which are all states where one player of team P is in the serving zone and all other players are on a central field in their half. As described above, $counter = -1$ in all serving states. Furthermore, there has not been any contact with the ball before a serve and the hardness is 0 since the ball rests and the opposing team has not determined the blocking player yet. The serving states S_Q^{serve} of team Q are defined analogously. Of course there exist more states which could be considered as serving states, e.g., serving from a corner which would be field $P01$. However, the listed states were the most common serving states, we observed when watching the videos of the beach volleyball tournaments of the Olympic games.

Besides the explicit naming of all serving states, all other states are also divided into four categories of states. This categorization helps when a decision rule is defined for this huge rally-SSO-MDP. Table 4.5 characterizes all categories of states. All categories are further differentiated according to the team controlling the ball. So, a state is, e.g. a receiving state for team Q if $counter = -1$, it is no state in $S_P^{serve} \cup S_Q^{serve}$ and the ball is on team Q 's court side.

The cardinality of the state space S can be computed from the cardinalities of the domains of the feature variables. It equals $24^4 \cdot 48 \cdot 5 \cdot 5 \cdot 3 \cdot 5$, which is more then $5.9 \cdot 10^9$ states. There is a small number of states in the state space that will not occur in a beach volleyball rally. For example, a combination of $counter = -1$ and $lastContact \neq \emptyset$ is not meaningful in a rally. The feature variable $counter = -1$ indicates that it is a serving situation and there should have been no contact by a player before this serving situation. Later in this section, it will be investigated how many states can be excluded due to non-meaningful combinations of feature variables.

Actions and Preconditions

The following action sets are defined only for team P . However, as already mentioned before, team Q 's action sets are defined analogously. In general, a team action consists of a specification of the next blocking player and two individual player actions. The player actions, in turn, consist of a hit and a movement. Depending on the current state, the available action sets differ. It is too much effort to specify the actions available in a state explicitly. Instead, A_s is defined as a factored set of feature actions together with preconditions on the feature actions which implicitly determine A_s . Let TA_P be a team action of team P . It consists of a specification of the blocking player and two player actions:

$$TA_P = (g_b, PA_{P1}, PA_{P2}).$$

The blocking player can be an arbitrary player of the team or not be specified, which is denoted by \emptyset :

$$dom(g_b) = \{P_1, P_2, \emptyset\}.$$

The player actions are themselves combinations of a hit with a movement:

$$PA = (tech_{target}, \mu).$$

The domain space of the *feature* actions are

$$\begin{aligned} dom(tech_{target}) &= dom(tech) \times dom(target) \times \{\emptyset\} \\ dom(tech) &= \{SF, SJ, r, r_m, s, FS, FE, FP, d, d_m\} \\ dom(target) &= \{P00, P01, \dots, P34, P35, Q00, Q01, \dots, Q34, Q35\} \\ dom(\mu) &= \{m_f, m_{fr}, m_r, m_{rb}, m_b, m_{bl}, m_l, m_{lf}, M_f, M_r, M_b, M_l, b, \emptyset\}. \end{aligned}$$

An explanation of the abbreviations for the different techniques, which constitute the domain space of *tech*, is contained in Table 4.6. The domain space of the target field equals the whole court grid. The domain space of μ contains all player movements. The movements abbreviated with *m* are one-field movements and the movements with *M* are movements over two fields. The direction of the movement is specified relative to the net by a subscript letter. The meanings of the subscript letters are: *f* = *forward*, *r* = *right*, *b* = *backward* and *l* = *left*. A blocking *b* is a special move since it is the only move which has a precondition. It belongs to the group of movements because a ball possession is not required to perform a block. All movements are compiled in Table 4.7.

The constitutional variables of a state must satisfy preconditions such that a feature action is available in the actions set of that state. The availability of a hit $tech_{target}$ is independently regulated from the availability of a movement μ through two different precondition functions. Later, functions are defined that consider the combinations of feature actions.

Let $tech_{target} \in dom(tech_{target})$ be a technique combined with a target field, then

$$\begin{aligned} prec_{hit} : S \times dom(tech_{target}) \times \{P_1, P_2, Q_1, Q_2\} &\rightarrow \{0, 1\} \\ (s, tech_{target}, \varrho) &\mapsto prec_{hit}(s, tech_{target}, \varrho) \end{aligned}$$

evaluates whether the hit $tech_{target}$ is available for player ϱ in s . If $prec_{hit}(s, tech_{target}, \varrho) = 1$, the $tech_{target}$ is available for player ϱ in s else not. All hitting techniques with their possible target fields and preconditions are listed in Table 4.6. For an efficient description of the preconditions, a function $neighbour(field)$ is used that returns the set of all neighboring fields of *field* according to the grid presented in Figure 4.6 including the field itself. In some states, the available action set of a player contains only the hit “no hit” which is denoted by \emptyset . Some examples picked from Table 4.6 are:

- A serving action is only allowed if *counter* = -1. So:

$$prec_{hit}(s, SF_{target}, \varrho) = 0 \forall \text{ player } \varrho, \forall \text{ target-fields } target \text{ if } counter \neq -1 \text{ in } s.$$

- A reception with a move, requires that the ball is in a neighbor field of the player:

$$\begin{aligned} prec_{hit}(s, r_{m,target}, P_1) &= 1 \forall \text{ target-fields } target, \\ &\text{if } pos(ball) \in neighbour(pos(P_1)) \text{ and } counter = -1 \text{ in } s. \end{aligned}$$

<i>tech</i>	<i>target</i>	Description	Preconditions <i>counter</i>	Position
\emptyset	-	no hit	none	none
Serve				
<i>SF</i>	$Q11 - Q24$	float serve	= -1	$pos(\varrho) = pos(ball) \in P01 - P04$
<i>SJ</i>	$Q11 - Q24$	jump serve (hard)	= -1	$pos(\varrho) = pos(ball) \in P01 - P04$
Reception				
<i>r</i>	$P11 - P34$	receive	= -1	$pos(ball) = pos(\varrho), s \notin S_P^{serve}$ ⁴
<i>r_m</i>	$P11 - P34$	receive with move	= -1	$pos(\varrho) \in neighbour(pos(ball)), pos(\varrho) \neq pos(ball), s \notin S_P^{serve}$
Setting				
<i>s</i>	$neighbour(pos(\varrho)) \setminus (Q, \cdot)$	set	> 0	$pos(\varrho) = pos(ball)$
Attack-Hit				
<i>FS</i>	$Q11 - Q24$	smash (hard)	> 1	$pos(\varrho) = pos(ball)$ or $pos(\varrho) + m_f = pos(ball)$
<i>FE</i>	$Q11 - Q24$	emergency shot	> 1	$pos(\varrho) \in neighbour(pos(ball))$
<i>FP</i>	$Q11 - Q34$	planned shot	> 0	$pos(\varrho) = pos(ball)$
Defense				
<i>d</i>	$P11 - P34$	defense	$\neq -1$	$pos(ball) = pos(\varrho)$
<i>d_m</i>	$P11 - P34$	defense with move	$\neq -1$	$pos(\varrho) \in neighbour(pos(ball)), pos(\varrho) \neq pos(ball)$

Table 4.6: Hit specification for player ϱ of team P and ball $ball$; requires always $\varrho \neq lastContact$ except the action *no hit*

The hitting techniques and preconditions of a player of team Q are defined analogously.

The only movement that has a precondition is the block. It can only be performed if the player $\varrho \in \{P_1, P_2\}$ is in a field close to the net which is a field in $\{P31, \dots, P34\}$. The precondition function for movements can be written compactly:

$$\begin{aligned}
 prec_{move} : S \times dom(\mu) \times \{P_1, P_2, Q_1, Q_2\} &\rightarrow \{0, 1\} \\
 (s, \mu, \varrho) &\mapsto \begin{cases} 0 & \text{if } \mu = b \wedge pos(\varrho) \notin \{P31, \dots, P34\} \text{ in } s \\ 1 & \text{else.} \end{cases}
 \end{aligned}$$

All possible movements of team P are listed in Table 4.7. The movements of the players that belong to team Q are defined analogously.

The model contains rules that restrict the possible combinations of a hit with a move to a player action as well as restrictions on the combination of two player actions to a team action. Reasons for these restrictions are general observations in real beach volleyball matches. The conditions are implemented by functions called *combc* which evaluate whether a combination of feature actions is allowed. The first function $combc_{PA}$ evaluates a combination of a hit with a movement:

$$\begin{aligned}
 combc_{PA} : (dom(tech_{target}) \cup dom(\mu)) &\rightarrow \{0, 1\} \\
 (tech_{target}, \mu) &\mapsto combc_{PA}(tech_{target}, \mu)
 \end{aligned}$$

The combinations for which $combc_{PA}(tech_{target}, \mu) = 0$ are:

⁴Forbids that a rally is started by a receive; is needed in the system dynamic to update counter.

Symbol	Specification	Description	Requirements
\emptyset	-	stay	none
m	$f, fr, r, rb, b, bl, l, lf$	move one field	none
M	f, r, b, l	move two fields	none
b	-	block	$pos(\varrho) \in \{P31, \dots, P34\}$, $counter \neq -1$

Table 4.7: Move specification for ϱ belonging to team P

1. If a player makes a real hit, i.e., a hit that is not “no hit” (\emptyset), only a one-field movement is allowed (due to timing reasons):

$$\text{If } tech_{target} \neq \emptyset \wedge \mu \in \{M_f, M_r, M_l, M_b\} \Rightarrow combc_{PA}(tech_{target}, \mu) = 0.$$

2. If a player makes a hit that includes a jump, i.e., a jump serve or a smash, only a one-field movement in forward direction (i.e., towards the net) is allowed to follow:

$$\text{If } tech \in \{SJ, FS\} \wedge \mu \notin \{m_f, \emptyset\} \Rightarrow combc_{PA}(tech_{target}, \mu) = 0.$$

3. If the hit requires a movement before executing the hit, no additional movement afterwards is allowed:

$$\begin{aligned} \text{If } tech \in \{r_m, d_m\} \wedge \mu \neq \emptyset &\Rightarrow combc_{PA}(tech_{target}, \mu) = 0, \\ \text{If } tech = FE \wedge pos(ball) \neq pos(\varrho) \wedge \mu \neq \emptyset &\Rightarrow combc_{PA}(tech_{target}, \mu) = 0. \end{aligned}$$

Furthermore, a restriction on the combination of player actions is incorporated into the model through the function

$$\begin{aligned} combc_{TA} : (dom(PA_{P1}) \cup dom(PA_{P2})) &\rightarrow \{0, 1\} \\ (PA_{P1}, PA_{P2}) &\mapsto combc_{TA}(PA_{P1}, PA_{P2}). \end{aligned}$$

The model contains only one restriction on the combination of player actions: If two player actions are combined to a team action, only one player may make a real hit:

$$\text{If } tech_{target} \neq \emptyset \text{ of } PA_{P1} \wedge tech_{target} \neq \emptyset \text{ of } PA_{P2} \Rightarrow combc_{TA}(PA_{P1}, PA_{P2}) = 0.$$

Furthermore, there exists a precondition when the designated blocking player ϱ_b of the team action may have a value unequal \emptyset . From the perspective of team P : Only if team P is the defending team and not in possession of the ball, i.e., if $side(pos(ball)) = Q$, the designated blocking player ϱ_b of the team action can be P_1 or P_2 . This holds analogously for a team action of team Q .

Team actions that themselves or whose player actions do not follow these rules are not available in the model – for both teams. Further conceivable restrictions could be easily implemented in the model whenever they only depend on the current state.

Transition Functions

The transition probabilities are determined by a transition function that is also decomposed into smaller probabilistic effects. This subsection uses a lot of notation and case distinctions to specify the transition probabilities in detail. The following might be very technical and difficult to read, however, it is necessary such that the rally-SSO-MDP can be implemented.

The transition function gets a state s and a feasible team action $TA_P \in \mathcal{A}$, of team P and a feasible team action $TA_Q \in \mathcal{A}'$, of the opponent team. Even if the opponent team is a part of the environment, a team action of the opponent team can be selected according to the fixed policy of team Q . The team action TA_Q of the opponent can be viewed as a disturbance that is incorporated in the transitions. The output of the transition function $trans$ is a probability distribution $\mathcal{P}(S)$ over the set of states:

$$\begin{aligned} trans : S \times \mathcal{A}_s \times \mathcal{A}'_s &\rightarrow \mathcal{P}(S) \\ (s_t, TA_P, TA_Q) &\mapsto trans(s_t, TA_P, TA_Q). \end{aligned}$$

The resulting probability distribution is derived from smaller probabilistic effects of the feature actions on the domain variables. In general, a conditional probability distribution specifies the probabilistic effect of a feature action on specific domain variables.

Example 4.1 (Conditional probability distribution):

For illustration, a short example of a probabilistic effect of a feature action is described: Consider the feature action $tech_{target}$ which is a hitting technique with a specified target field. Under the condition that $tech_{target}$ is performed by player P_1 of team P (abbreviated by $tech_{target} \in PA_{P_1}$) from field $field = pos(P_1)$ in the current state s and the opponent's team action does not include a blocking action (abbreviated by $b \notin TA_Q$), the probability that the domain variable $pos(ball)$ equals $target$ in the next state is determined by the individual success probability of the hitting player P_1 :

$$\mathbb{P} \{ pos(ball) = target \mid tech_{target} \in PA_{P_1}, field = pos(P_1), b \notin TA_Q \} = p_{succ, P_1}(field, tech_{target}). \quad *$$

A decomposition in probabilistic effects only works if the decomposed effects are independent. If there exists a correlation between the effects of several feature actions, one would have to specify a joint conditional probability distribution for that set of correlated feature actions. However, this rally-SSO-MDP is designed on the basis of, so-called, individual player skills which are assumed to be independent probabilistic effects. So, the decomposition of the transition probabilities into smaller probabilistic effects is beneficial.

This definition of the transition probabilities is similar to that used by the Relational dynamic influence diagram language (RDDL), described in Sanner (2010). In an RDDL-representation, for each action and each domain variable, a conditional probability distribution over the values of that variable in the next state is specified. This representation is useful if many objects evolve independently and simultaneously.

In the following, a list of all probabilities on which the independent probabilistic effects are based is presented. These probabilities are required as input probabilities for the rally-SSO-MDP. As they depend only on the skills of a single player, they are called *individual player skills*.

Assume, for each player g being on any position $field$ and for each hitting technique $tech_{target}$ the success probability

$$p_{succ, g}(field, tech_{target}) := \mathbb{P}(pos^{t+1}(ball) = target \mid field, tech_{target})$$

is known. It is the probability that the specified target field $target$ from g 's position is met. In the notation used above, the terms $field$ and $tech_{target}$ show the dependence the player's position and the hit he uses. The probability is time-independent. The t on the right-hand side of the last equation is only used to indicate that $pos^{t+1}(ball)$ is the position of the ball in the subsequent state.

Similar, assume for each player g using any hit $tech_{target}$ from any position $field$ the probability of an execution fault

$$p_{fault,g}(field, tech_{target}) := \mathbb{P}(s^{t+1} = fault \mid field, tech_{target})$$

is known. An execution fault includes hits where the ball is not correctly hit such that the referee terminates the rally with a fault for the hitting player. For serves and attack-hits an execution fault also includes that the ball is hit into the net. In addition to a successful hit and an execution fault, the model also contains the possibility of a "nearly successful hit" or a *deviation*. A deviation is a hit that lands in a neighbor field of the target field. Of course, the deviation can result in an outside-field if the original target field is a border field. The remaining probability

$$1 - p_{succ,g}(field, tech_{target}) - p_{fault,g}(field, tech_{target}) =: p_{dev,g}(field, tech_{target})$$

is the probability that the ball lands in a neighboring field of the target field. It is assumed that each neighboring field of the target field is equally probable.

For the defending techniques r , r_m , d and d_m , it is also distinguished between a reception or defense of a hard and a normal ball. So, for those techniques, the individual player probabilities do also depend on the hardness of the ball in the current state. Since this dependence does not apply to all types of hits, it is denoted in brackets, i.e., $p_{succ,g}(pos^t(g), tech_{target}, [hardness])$.

Furthermore, assume that the blocking skills of each player are known. The parameter $p_{g,block}$ denotes the probability that player g touches the ball when performing the block b against an adequate attack-hit from the opponent's side of the court. The probability $p_{g,block}$ is independent of the skills of the attacking player and should be measured against an average attack hit. There are three possible outcomes of a blocking player who gets the ball. The block can be strong such that it is impossible for the opposing team to defend the returned ball, and the blocking team wins the rally. This probability is denoted by $p_{g,block,point}$. Furthermore, the block can result in a fault with probability $p_{g,block,fault}$. This happens if the ball is blocked into the net and cannot be regained or the blocking player touches the net, which is an execution fault. None of the above happens with probability $p_{g,block,ok} := p_{g,block} - p_{g,block,point} - p_{g,block,fault}$. This is called an "ok"-block, and the ball lands in a random field on the court. It is assumed that each field on both sides of the net is equally probable. The probability that the blocking player fails to get his hands on the ball is defined as $p_{g,no\ block} := 1 - p_{g,block}$. In this case, the direction of the ball and its landing field is not affected by the block. In total, the blocking probabilities are

$$p_{g,no\ block} + \underbrace{p_{g,block,point} + p_{g,block,fault} + p_{g,block,ok}}_{p_{g,block}} = 1.$$

After having specified all input probabilities, it is explained how the independent decomposed effects are combined, and the next state is determined. The transition function is defined as a decomposition of three functions:

$$trans(s_t, TA_P, TA_Q) = (trans_{det} \circ trans_{block} \circ trans_{hit})(s_t, TA_P, TA_Q)$$

with

$$\begin{aligned} trans_{hit} &: S \times A \times A' && \rightarrow S \times A \times A' \times \mathcal{P}(S) \\ trans_{block} &: S \times A \times A' \times \mathcal{P}(S) && \rightarrow S \times A \times A' \times \mathcal{P}(S) \\ trans_{det} &: S \times A \times A' \times \mathcal{P}(S) && \rightarrow \mathcal{P}(S). \end{aligned}$$

In the function declaration above, S is the state set, A is the set of all team actions of team P , A' the set of all team actions of team Q and $\mathcal{P}(S)$ the set of all probability distribution over S . As the names of the transition functions suggest, the $trans_{hit}$ captures the probabilistic effect related to the hitting skills and $trans_{block}$ the probabilistic effect related to the blocking skills. The $trans_{det}$ function includes all deterministic transitions that follow from the results of $trans_{hit}$ and $trans_{block}$ like, e.g., the new player-positions or increments of the *counter*. The final result is a probability distribution over the set of states. The intermediate probability distributions may be coarse probability distributions, i.e., they only assign probabilities to sets of states. However, in each function, the probability distribution becomes more and more detailed such that at the end a probability is defined for each state. The original state and the team actions are also passed to the next function such that the necessary information for the transition is available in each of the three functions.

The first probabilistic effect comes through the hitting skills into the system dynamics and specifies the ball's next position. The resulting probability distribution consists of four sets, which are named S_{succ} , S_{dev} , S_{fault} and S_{point} . These are the only sets to which a probability greater than zero may be assigned. The state set S_{fault} will either consists only of the state *fault* or the state *point*. If no hitting player exists or the hitting player is of team P , $S_{fault} := \{fault\}$. If the hitting player is of team Q , $S_{fault} := \{point\}$. Accordingly, the set S_{point} contains the *point* or *fault*. In contrast to the absorbing states *fault* and *point*, which are defined from the perspective of team P , the sets S_{point} and S_{fault} are always defined from the perspective of the hitting team. The state set S_{succ} and S_{dev} may contain a large number of possible subsequent states. However, they may also be empty. In the following, it is explained how these sets are defined and which probabilities are assigned to them.

Assume that one of the team-actions TA_P or TA_Q contains a real hit $tech_{target}$ and $g \in \{P_1, P_2, Q_1, Q_2\}$ is the hitting player. As only feasible team actions are considered, there is – due to the preconditions and the combination conditions – at most one player who is hitting the ball. The function $trans_{hit}$ first checks the *double-contact-rule* and the *four-hits-rule*. This means, if $lastContact = g$ or $counter = 3$ in the current state, the resulting probability distribution regarding the subsequent state s_{t+1} is

$$\mathbb{P}\{s_{t+1} \in S_{fault}\} := 1.$$

In this case, the sets S_{succ} and S_{dev} are not further specified and empty sets, and the probability of all other sets different from S_{fault} is zero.

If no rule is violated, the probability distribution over the state sets S_{succ} , S_{dev} and S_{fault} is defined according to the individual skills of g :

$$S_{succ} := \{s_{t+1} \in S \mid pos^{t+1}(ball) = target \text{ in } s_{t+1}\}$$

$$\mathbb{P}\{s_{t+1} \in S_{succ} \mid \xi \text{ plays } tech_{target}\} = p_{succ,\xi}(pos^t(\xi), tech_{target}, [hardness]),$$

$$S_{dev} := \{s_{t+1} \in S \mid pos^{t+1}(ball) \in neighbour(target) \setminus \{target\} \text{ in } s_{t+1}\}$$

$$\mathbb{P}\{s_{t+1} \in S_{dev} \mid \xi \text{ plays } tech_{target}\} = p_{dev,\xi}(pos^t(\xi), tech_{target}, [hardness]),$$

$$\mathbb{P}\{s_{t+1} \in S_{fault} \mid \xi \text{ plays } tech_{target}\} = p_{fault,\xi}(pos^t(\xi), tech_{target}, [hardness]).$$

The parameters $pos^t(\xi)$ and $hardness$ are the domain variables of the current state s_t . Observe, that S_{succ} respectively S_{dev} contain quite a number of states since besides $pos(ball)$ no other domain variable is specified. In the state set S_{dev} , the position of the ball may be any neighbor field of the target field but not the target field itself. The dependency of the individual player skills on the hardness of the ball in the current state is denoted in brackets and applies only to receptions and defenses.

If neither TA_P nor TA_Q contains a real hit either S_{point} or S_{fault} is assigned a probability of 1 depending on the current position of the ball and the last contact. As an abbreviation for *ball is on Team P's side of the net* the term $pos(ball) \in side(P)$ is used. Furthermore, $pos(ball) \in out$ denotes that the ball is in an outside field. With this abbreviations, the outcome of $trans_{bit}$ when no player is hitting can be specified as:

$$\mathbb{P}\{s_{t+1} \in S_{fault} \mid \text{nobody hits}\} = \begin{cases} 1 & \text{if } pos(ball) \in out \wedge lastContact \in \{P_1, P_2\}, \\ 1 & \text{if } pos(ball) \notin out \wedge pos(ball) \in side(P), \\ 0 & \text{else,} \end{cases}$$

$$\mathbb{P}\{s_{t+1} \in S_{point} \mid \text{nobody hits}\} = \begin{cases} 1 & \text{if } pos(ball) \in out \wedge lastContact \in \{Q_1, Q_2\}, \\ 1 & \text{if } pos(ball) \notin out \wedge pos(ball) \in side(Q), \\ 0 & \text{else.} \end{cases}$$

In this case, all other sets are empty and have zero probability.

All cases related to a potential hitting action were now considered. The outcome of $trans_{bit}$ is a definition of the state sets S_{succ} , S_{dev} , S_{fault} and S_{point} together with a specified probability assigned to each of the sets. As in the next step the probabilities of these sets are redefined, it is denoted by $\mathbb{P}_{trans_{bit}}\{\cdot\}$ the probability distribution defined by $trans_{bit}$.

In the next step, this probability distribution is further refined by the function $trans_{block}$. As a blocking action is classified as a movement, there may be more than one player who blocks. Observe that the domain variable $blockingPlayer$ does not pretend the blocking player, its only purpose is that a team can specify a defense strategy that relies on the same designated blocking player over more than one decision epoch. For all players that perform a blocking action, it is evaluated in the first step whether their block may have an impact on the ball. A block must be performed in a field at a net to have an impact. This is satisfied by all blocking player as only feasible team actions are considered. Also, the blocking player has to face the hitting player. So, if the hitting player ξ is in field P_{ij} [Q_{ij}] on the court, the blocking player ξ_b must be on the opposed side of the net in field $Q3(5-j)$ [$P3(5-j)$]. The function $blockMayHaveImpact$ specifies this condition on the block by using the current state s_t and the current positions of the hitting player ξ and the blocking player ξ_b

$$blockMayHaveImpact(s_t, \xi, \xi_b) = \begin{cases} 1 & col(pos^t(\xi)) = col(pos^t(\xi_b)) \wedge side(pos^t(\xi)) \neq side(pos^t(\xi_b)) \\ 0 & \text{else.} \end{cases}$$

The function $col(field)$ does return the second index variable of a field according to numbering in the grid. The only case where the block of two players may have an impact at the same time is when both players are positioned in the same field at the net facing the hitting player. In this case, the blocking skills of the two players are mixed with a factor 0.5 which can be interpreted as if every player comes to a block with a probability of 0.5. In the following, assume g_b is the only blocking player who may be, in the case that two players blocking in the same field, imagined as an artificial player with the combined skills of both players.

The blocking skills of player g_b are used to further specify the probability distribution. With probability $p_{g_b, no\ block}$ the blocking player g_b misses the ball and the flight trajectory of the ball does not change. Therefore, the probability of S_{succ} and S_{dev} is multiplied with $p_{g_b, no\ block}$. The result is the probability that the hit is successful (respectively a deviation) and the balls flight trajectory was not changed by the block:

$$\begin{aligned}\mathbb{P}\{s_{t+1} \in S_{succ}\} &:= \mathbb{P}_{trans_{hit}}\{s_{t+1} \in S_{succ}\} \cdot p_{g_b, no\ block}, \\ \mathbb{P}\{s_{t+1} \in S_{dev}\} &:= \mathbb{P}_{trans_{hit}}\{s_{t+1} \in S_{dev}\} \cdot p_{g_b, no\ block}.\end{aligned}$$

The blocking player makes with probability $p_{g_b, block, point}$ a block that can not be successfully defended. This results in a direct point for the blocking team which is a fault for the hitting team. So, the probability of S_{fault} is updated by:

$$\mathbb{P}\{s_{t+1} \in S_{fault}\} := \mathbb{P}_{trans_{hit}}\{s_{t+1} \in S_{fault}\} + (\mathbb{P}_{trans_{hit}}\{s_{t+1} \in S_{succ}\} + \mathbb{P}_{trans_{hit}}\{s_{t+1} \in S_{dev}\}) \cdot p_{g_b, block, point}.$$

The blocking team makes a fault and the hitting team gains a point if the block is a bad block. A bad block hits the ball into the net or into an outside field. Also the block may be executed in a forbidden way such that the referee indicates a fault which may happen if, e.g., the blocking player touches the net. So, the probability of the state set S_{point} is updated by

$$\mathbb{P}\{s_{t+1} \in S_{point}\} := \mathbb{P}_{trans_{hit}}\{s_{t+1} \in S_{point}\} + (\mathbb{P}_{trans_{hit}}\{s_{t+1} \in S_{succ}\} + \mathbb{P}_{trans_{hit}}\{s_{t+1} \in S_{dev}\}) \cdot p_{g_b, block, fault}.$$

Due to the blocking action, a new set of states with a positive probability arises. If a block is neither a point nor a fault but the ball is blocked, it is called an *ok-block*. As it is not distinguish between different blocking directions, the resulting field from an ok-block is just a random field of the court without the outside fields. Define $S_{rand} := \{s \in S \mid pos(ball) \in \{P11 \dots P34, Q11 \dots Q34\}\}$, then

$$\mathbb{P}\{s_{t+1} \in S_{rand}\} := (\mathbb{P}_{trans_{hit}}\{s_{t+1} \in S_{succ}\} + \mathbb{P}_{trans_{hit}}\{s_{t+1} \in S_{dev}\}) \cdot p_{g_b, block, ok}.$$

The reader may observe that the probabilities of all specified sets still sum up to 1. If there is no player that performs a block that may have an impact on the ball, the $p_{g_b, no\ block}$ is set to 1 and in doing so the probabilities of S_{succ} , S_{dev} , S_{fault} and S_{point} stay unchanged.

There is another edge case that is treated appropriately: If the result of the hitting action was for sure a state in S_{fault} , the blocking action does not change this already completed rally. Namely, in this case $\mathbb{P}_{trans_{hit}}\{s_{t+1} \in S_{succ}\}$ and $\mathbb{P}_{trans_{hit}}\{s_{t+1} \in S_{dev}\}$ is zero, such that the modification with the blocking skills has no impact.

After the probabilistic effects of hits and blocks are incorporated into the model, the $trans_{det}$ function evaluates all remaining transitions. Since the potential subsequent states are compiled in sets S_{fault} , S_{succ} ,

S_{dev} and S_{rand} , all remaining transitions are deterministic – conditioned by the set in which the states are contained. The $trans_{det}$ function includes transitions that determine the new players' positions and the values of the domain variables *counter*, *lastContact*, *hardness* and *blockingPlayer*. It first reduces the state sets S_{succ} , S_{dev} and S_{rand} to states that have values according to the deterministic transitions and then specifies the final probability distribution. In the following, it is explained, which values the domain variables of states in S_{succ} , S_{dev} or S_{rand} should have:

A movement which is not a block is no probabilistic effect since it is assumed to be always successful. As the list of all possible outcomes of movements starting from different fields is very long, an auxiliary function

$$makeStep : \{P00, P01, \dots, Q34, Q35\} \times dom(\mu) \rightarrow \{P00, P01, \dots, Q34, Q35\}$$

$$(field_1, \mu) \mapsto field_2$$

is defined. It returns the field $field_2$ resulting from a movement μ starting from $field_1$. Field $field_2$ is selected according to the grid presented in Figure 4.6 and the specified direction in μ . For example,

$$makeStep(P21, m_{lf}) = P32 \text{ and } makeStep(Q33, M_b) = Q13.$$

If a movement goes beyond the fields specified in the grid, the last field in the desired direction is returned. To distinguish between positions in the current state s_t and positions in the subsequent state s_{t+1} , pos^t is used for the current position and pos^{t+1} for the positions belonging to the subsequent state s_{t+1} . For all states remaining in S_{succ} , S_{dev} or S_{rand} , it should hold for the position of the non-hitting player g :

$$pos^{t+1}(g) = makeStep(pos^t(g), \mu_g).$$

For the hitting player, the step starts at the old position of the ball. This is relevant as hits like r_m or d_m contain movements prior to the hit. So, the new position of the hitting player g is

$$pos^{t+1}(g) = makeStep(pos^t(ball), \mu_g).$$

If no real hit is contained in the team actions, the sets S_{succ} and S_{dev} are already empty. So, assume player g performs a real hit $tech_{target}$. Then, from the sets S_{succ} and S_{dev} , all states with $lastContact = g$ are selected. Furthermore, only states where the counter is incremented in a correct way, stay in the set S_{succ} and S_{dev} . Depending on the hitting technique, it is distinguished whether the ball crossed the net after a serve or a field attack. In the first case, only states with $counter = -1$ may stay in the sets, in the latter case only states with $counter = 0$. If the ball did not cross the net, the counter is incremented by 1. However, after a receiving situation, the counter is set from -1 to 1. The following table gives an overview over the described new values of *counter*:

$side(pos^{t+1}(ball)) \neq side(pos^t(ball))?$	<i>tech</i>	$counter^{t+1}$
yes	$\in \{SJ, SF\}$	-1
yes	$\notin \{SJ, SF\}$	0
no	$\in \{r, r_m\}$	1
no	$\notin \{r, r_m\}$	$lastContact^t + 1$

The set S_{rand} is handled similar: If no adequate blocking action is contained in the team actions, the set S_{rand} is empty. Assume there exists an adequate block executed by player g_b . Let s_t be the current state of the rally-SSO-MDP which was passed to the $trans$ function. Then, all states $s_{t+1} \in S_{rand}$ with $lastContact = g_b^5$ and $counter = 1$ are selected by $trans_{det}$ if the ball was played over the net, i.e., if $side(pos^{t+1}(ball)) \neq side(pos^t(ball))$. This is the case when the blocking player has touched the ball and maybe changed the ball's flight trajectory, but the ball crossed the net. If $side(pos^{t+1}(ball)) = side(pos^t(ball))$, all states with $lastContact = \emptyset$ and $counter = 0$ may remain in the set S_{rand} . This corresponds to the case, when the ball is blocked back to the attacking team's side. The following table, characterizes all combinations of $counter$ and $lastContact$ that remain in the state set S_{rand} :

$side(pos^{t+1}(ball)) \neq side(pos^t(ball))?$	$counter^{t+1}$	$lastContact^{t+1}$
yes	1	g_b
no	0	\emptyset

At this point, we see that it is not possible to evaluate the $trans_{det}$ function prio to the other transition functions. Depending on the considered set, the values of the deterministic domain variables differ. For example, if the ball crossed the net after a field attack, it depends on whether the state is from S_{rand} or $S_{succ} \cup S_{dev}$ to determine the last contact.

Finally, the parameters $hardness$ and $blockingPlayer$ are determined and all states in S_{rand} , S_{succ} and S_{dev} that contain other values are removed. If the hitting player executed a smash FS or a jump serve SJ , the domain variable $hardness$ is set to $hard$ and otherwise to $normal$:

$$hardness := \begin{cases} hard & \text{if } tech \in \{FS, SJ\} \\ normal & \text{else.} \end{cases}$$

If the team action of the defending team specifies a new designated blocking player, i.e., g_b is unequal to \emptyset , the domain variable $blockingPlayer$ is set to that player. If the ball changes the courtside, the $blockingPlayer$ is automatically reset to \emptyset .

In the last step $trans_{det}$ specifies the final probability distribution. For each state s in each state set $\tilde{S} \in \{S_{succ}, S_{dev}, S_{fault}, S_{point}, S_{rand}\}$ the probability is computed as

$$\mathbb{P}\{s_{t+1} = s \mid s \in \tilde{S}\} := \begin{cases} \frac{1}{|\tilde{S}|} & \text{if } \mathbb{P}_{trans_{block}}\{\tilde{S}\} > 0 \\ 0 & \text{else.} \end{cases}$$

For all other states, not being in any of the sets, the probability is set to zero. Observe that this is the first probability distribution that specifies a probability for each single state. After the elimination of states in these sets by the $trans_{det}$ function, the intersection of the sets S_{succ} , S_{dev} , S_{fault} , S_{point} , S_{rand} is empty. So, the probability of each subsequent state is well defined. Furthermore, all probabilities sum up to 1.

The reward structure of the SSO-MDP is defined according to the definition: All transitions have zero rewards except the transition to the state $point$, which has reward 1.

The presented SSO-MDP satisfies Assumption 3.2.1: For hits in a beach volleyball match, it is reasonable to assume that $p_{fault,g}(field, tech_{target}) > 0$ holds. Since an execution fault directly leads to a fault, the SSO-MDP assumption follows if $p_{fault,g}(field, tech_{target}) > 0$ holds for all hits.

⁵In the special case of two player blocking in the same field, the set S_{rand} may contain all states where $lastContact$ equals one of both players.

However, this is only mainly supported by the data presented in Subsection 4.3.3. For example, the float serves of Julius Brinks estimated from observations of the final have a zero fault probability, compare Table A.1. In comparison in Table 4.12, the fault probability of a float serve of Julius Brink is greater than zero. This supports the statement that $p_{\text{fault},g}(\text{field}, \text{tech}_{\text{target}}) > 0$ is reasonable for hits in a beach volleyball rally if the estimates are based on a large enough number of observations.

4.3.2 Defining a Decision Rule

In an SSO-MDP, an optimal stationary policy is defined by a decision rule. A decision rule specifies for each state a probability distribution over the set of available actions, see Definition 2.1.2. In large MDPs with a large number of states, it may get complicated to specify a probability distribution for each state explicitly. Furthermore, when different decision rules should be compared, it may be hard to overlook the differences between the decision rules under considerations. Nevertheless, a specification of a probability distribution for the next action choice in each state is necessary to be able to compare different decision rules.

Assume, different serving techniques (float serve versus jump serve) and different techniques for a field attack (smash versus shot) should be compared in the rally-SSO-MDP. A simple approach for specifying decision rules that allow comparing those techniques is:

- In states where a choice between the hitting techniques under considerations is made, specify a probability distribution that represents the decision rule under investigation.
- In all other states, choose evenly distributed an arbitrary action from the set of available actions.

If the rally-SSO-MDP is simulated over 1000 sets using such a decision rule, the characteristics of the resulting rallies do not match those observed from a real beach volleyball rally. In a simulation⁶ of the rally-SSO-MDP with real skill estimates but using the described decision rule, the average number of field attacks of one team in a set is 2.6 with a standard deviation of 1.5. These numbers are too low in comparison with statistics of real beach volleyball matches. For instance, Giatsis and Panagiotis collected data of 118 sets from the first 2003 FIVB men's beach volleyball tournament in Rhodes, Greece and got on average around 40 attacks per team for a match with two sets (Giatsis and Panagiotis, 2008). Another example is the match recap of the Olympic Games Gold Medal Match of 2012, where 64 attacks by Germany and 74 by Brazil were made over 3 sets (Database, 2018). So, a decision rule that just uniformly randomizes over the set of available actions provides too little coordination between the players and also bad positioning on the field such that services cannot be received or field attack attempts are poorly coordinated.

Having these observations in mind, a concept was *parametrized basic decision rule* developed. It should be a trade-off between guaranteeing reasonable play dynamics and focusing on the differences of the decision rules under consideration. First, the concept of a *basic decision rule* is explained which is afterward extended by a parametrization.

The basic decision rule is a decision rule that specifies a probability distribution over the set of available actions so precisely such that the characteristics of a beach volleyball set are kept. It should exclude unrealistic and non-optimal combinations of player actions. At the same time, the decision rule

⁶Screenshots of the simulation can be found in Appendix B and the simulation itself in the supplementary material provided in Appendix E.

tries to be as less specific as possible. This is ensured by choosing the actions equally distributed over the set of *reasonable actions*. In the rally-SSO-MDP, the hit *no hit* and the move *stay* were included such that in each state it is possible that both players do nothing. It is assumed, that this default action, which is always available in \mathcal{A}_s , is included in R_s if no other action is considered as reasonable. With this assumption, the basic decision rule can be defined as follows:

Definition 4.3.1 (Basic decision rule).

Let $R_s \subseteq \mathcal{A}_s$ be a set of reasonable actions with $|R_s| > 0$. Then, the basic decision rule with regard to R_s is defined as

$$d_{basic} : S \rightarrow \mathcal{P}(R_s),$$

$$s \mapsto q_{d(s)}(\cdot) \text{ s.t. } q_{d(s)}(a) = \frac{1}{|R_s|}.$$

In the rally-SSO-MDP, each state is classified according to the categories: serving state, receiving state, defending state, setting state or attacking state, see Table 4.5. The defined basic decision rule for the rally-SSO-MDP selects a team action of the *reasonable action set*. The *reasonable action sets* are based on the class of the state, see Appendix C for more details on the basic decision rule of the rally-SSO-MDP. There is one change in the basic decision rule in comparison to the basic decision rule used in Hoffmeister and Rambau (2017b). After a serve the hitting player makes a one-field movement towards the net. This modification is reasonable from a practitioner's view and yields a slightly better fit to the realized probabilities, which can be seen in the next subsection.

If the rally-SSO-MDP is again simulated over 1000 sets using the same skill estimates, but this time the basic decision rule as specified in Appendix C, the average number of field attacks per team per set is 31 with a standard deviation of 6. This number fits better to the statistics from real matches presented above. It may be a bit too large, which could be a hint that the players are too well coordinated.

The reader may ask whether a basic decision rule and the preconditions on the available actions could be combined. The answer is that preconditions contain negative characteristics of actions which restrict the set of available actions while the basic decision rule specifies positive characteristics of actions that should be chosen in each state with a positive probability. Of course, it would be possible to exclude in the preconditions all actions that are not positively characterized in the basic decision rule. However, this would be an unfavorable way.

A parametrization of the basic decision rule is included in states where the question of decision plays a role. At those states, a parameter modifies the uniform distribution. However, still, only reasonable actions may be chosen. If the question of interest is a binary decision like, e.g., a float serve or a jump serve, a border field or a non-border field, the parametrization can be implemented by using a single parameter in $[0, 1]$. The following definition specifies what is considered as a binary parametrization of the basic decision rule:

Definition 4.3.2 (Binary parametrization of the basic decision rule).

Consider a basic decision rule d_{basic} with R_s being the set of reasonable actions in state $s \in S$. A binary question specifies two distinguished subsets R_s^1 and R_s^2 of R_s with $R_s^1 > 0$ and $R_s^2 > 0$ that each

contain one variant of the binary property under consideration. Then, a binary parametrization of the basic decision rule is implemented by a single parameter $\pi \in [0, 1]$:

$$q_{d(s)}(a) = \begin{cases} \frac{1-\pi}{|R_s^1|} & \text{if } a \in R_s^1 \\ \frac{\pi}{|R_s^2|} & \text{if } a \in R_s^2 \\ 0 & \text{else.} \end{cases}$$

Definition 4.3.2 specifies a valid probability distribution over R_s : For each $a \in R_s$, it holds $q_{d(s)}(a) \geq 0$ and

$$\sum_{a \in R_s} q_{d(s)}(a) = \sum_{a \in R_s^1} q_{d(s)}(a) + \sum_{a \in R_s^2} q_{d(s)}(a) = |R_s^1| \cdot \frac{1-\pi}{|R_s^1|} + |R_s^2| \cdot \frac{\pi}{|R_s^2|} = 1.$$

Observe that if $R_s^1 \cup R_s^2 = R_s$ holds, the basic decision rule can be regained by setting $\pi = \frac{|R_s^2|}{|R_s|}$. This is noted in the following observation:

Proposition 4.3.3:

Let d_{basic} be the basic decision rule with regard to the reasonable action sets R_s , $s \in S$. Let R_s^1 and R_s^2 be a partition of R_s , that is parametrized by $\pi \in [0, 1]$ according to Definition 4.3.2.

Then, setting $\pi = \frac{|R_s^2|}{|R_s|}$ regains the basic decision rule.

PROOF. Since R_s^1 and R_s^2 are a partition of R_s , for each $a \in R_s$ it either holds $a \in R_s^1$ or $a \in R_s^2$. Assume, $a \in R_s^1$ holds. Then,

$$q_{d(s)}(a) = \frac{\left(1 - \frac{|R_s^2|}{|R_s|}\right)}{|R_s^1|} = \frac{|R_s| - |R_s^2|}{|R_s| \cdot |R_s^1|} = \frac{|R_s^1|}{|R_s| \cdot |R_s^1|} = \frac{1}{|R_s|}.$$

Assume, $a \in R_s^2$ holds. Then,

$$q_{d(s)}(a) = \frac{\frac{|R_s^2|}{|R_s|}}{|R_s^2|} = \frac{|R_s^2|}{|R_s| \cdot |R_s^2|} = \frac{1}{|R_s|}.$$

So, the resulting probability distribution corresponds to that of the basic decision rule. ■

Similar to the example question in the set-SSO-MDP of Section 4.2, the following set of example questions has been implemented in the rally-SSO-MDP by binary parameterizations of the basic decision rule:

- Should player 2 or player 1 be the blocking player? (π_b)
- Should a float serve or a jump serve be performed by player ϱ ? ($\pi_{b,tech}^{serve}(\varrho)$)
- Should the service be made towards a non-border field or a border field by player ϱ ? ($\pi_{b,field}^{serve}(\varrho)$)
- Should a shot or a smash be performed by player ϱ ? ($\pi_{b,tech}^{field}(\varrho)$)

- Should the field attack be made towards a non-border field or a border field by player g ? ($\pi_{b,field}^{field}(g)$)
- Should the service be made on opponent player 2 or opponent player 1? (π_s)

The parameters used for modeling the questions are specified in brackets behind the corresponding question. Let for each question, R_s^1 always be the set of reasonable actions for which the property first mentioned in the question holds. As a mnemonic, this is for questions regarding a hitting situation always the safer opportunity. According to Definition 4.3.2, the value of the binary parameter equals the probability of the more risky opportunity for hitting decisions. For example,

parameter	value
π_b	0
$\pi_{b,tech}^{serve}(P_1)$	0
$\pi_{b,field}^{serve}(P_1)$	0
$\pi_{b,tech}^{field}(P_1)$	1
$\pi_{b,field}^{field}(P_1)$	1
$\pi_{b,tech}^{serve}(P_2)$	0
$\pi_{b,field}^{serve}(P_2)$	0
$\pi_{b,tech}^{field}(P_2)$	1
$\pi_{b,field}^{field}(P_2)$	1
π_s	0

is a parametrization of the basic decision rule, where player 2 is always the blocking player, both players use a float serve towards a non-border field and a smash towards a border field, and a service is made always on the opponent player 2.

As in the example questions, it may happen that several binary question concern the same states. For instance, $\pi_{b,tech}^{serve}(P_1)$ and $\pi_{b,field}^{serve}(P_1)$ both concern states where player P_1 is serving. A simultaneous implementation of binary questions is defined as a straight forward generalization of the binary parametrization:

Definition 4.3.4 (Simultaneous binary parametrization of the basic decision rule).

Consider the basic decision rule d_{basic} with regard to the reasonable action sets R_s , $s \in S$.

Assume n binary question that concern the same state s in S . Each binary question $i \in \{1, \dots, n\}$ specifies two distinguished subsets $R_s^1(i)$ and $R_s^2(i)$ of R_s , each containing a variant of the binary property under consideration such that $|R_s^1(i)| > 0$ and $|R_s^2(i)| > 0$ holds. Then, a simultaneous binary parametrization of the basic decision rule is implemented by n parameters $\pi_i \in [0, 1]$, $i \in [n] := \{1, \dots, n\}$ as follows: For each $a \in \mathcal{A}_s$ define an index set $J_a \subseteq [n]$ such that

$$J_a := \begin{cases} \emptyset & \text{if } \exists i : a \notin R_s^1(i) \wedge a \notin R_s^2(i) \\ \bigcap_{i:a \in R_s^1(i)} \{i\} & \text{else.} \end{cases}$$

Using that index set J_a , the basic decision rule is defined as

$$q_{d(s)}(a) = \frac{\prod_{i \in J_a} (1 - \pi_i) \cdot \prod_{i \in [n] \setminus J_a} \pi_i}{|\bigcap_{i \in J_a} R_s^1(i) \cap \bigcap_{i \in [n] \setminus J_a} R_s^2(i)|}$$

for a with $J_a \neq \emptyset$ and else as $q_{d(s)}(a) = 0$.

Analogously to a single binary parametrization, we can convince ourselves that the specified probabilities defines a probability distribution over R_s : All specified probabilities are non-negative and we have

$$\begin{aligned}
& \sum_{a \in R_s} q_{d(s)}(a) \\
&= \sum_{J \subseteq [n]} \sum_{\substack{a \in R_s: \\ a \in R_s^1(i), \forall i \in J \\ \wedge a \in R_s^2(i), \forall i \in [n] \setminus J}} \frac{\prod_{i \in J} (1 - \pi_i) \cdot \prod_{i \in [n] \setminus J} \pi_i}{|\bigcap_{i \in J} R_s^1(i) \cap \bigcap_{i \in [n] \setminus J} R_s^2(i)|} \\
&= \sum_{J \subseteq [n]} \left[\prod_{i \in J} (1 - \pi_i) \cdot \prod_{i \in [n] \setminus J} \pi_i \right] \\
&= \sum_{J \subseteq [n-1]} \left[\prod_{i \in J} (1 - \pi_i) \cdot \prod_{i \in [n-1] \setminus J} \pi_i \cdot ((1 - \pi_n) + \pi_n) \right] \\
&= \sum_{J \subseteq [n-1]} \left[\prod_{i \in J} (1 - \pi_i) \cdot \prod_{i \in [n-1] \setminus J} \pi_i \right] \\
&= \dots = (1 - \pi_1) + \pi_1 = 1
\end{aligned}$$

In the case of simultaneous binary parametrization, it is also possible to regain the basic decision rule if all questions i partition the complete set R_s into R_s^1 and R_s^2 .

Proposition 4.3.5:

Let d_{basic} be the basic decision rule with regard to the reasonable action sets R_s , $s \in S$. Let $R_s^1(i)$ and $R_s^2(i)$ be partitions of R_s according to question i , $i \in \{1, \dots, n\}$. Assume n parameters $\pi_i \in [0, 1]$, $i \in [n] := \{1, \dots, n\}$ that simultaneously parametrize the basic decision rule according to Definition 4.3.4.

Then, choosing for all $i \in [n]$ the parameter π_i such that

$$\prod_{i \in J} (1 - \pi_i) \cdot \prod_{i \in [n] \setminus J} \pi_i = \frac{|\bigcap_{i \in J} R_s^1(i) \cap \bigcap_{i \in [n] \setminus J} R_s^2(i)|}{|R_s|}$$

regains the basic decision rule.

PROOF. Let a be an action in R_s . Since all questions partition the complete set R_s , there must exist a $J \subseteq [n]$ with $a \in R_s^1(i)$, $\forall i \in J$ and $a \in R_s^2(i)$, $\forall i \in [n] \setminus J$. Define π_i according to the proposition. Then the probability of action a is

$$\begin{aligned}
& q_{d(s)}(a) \\
&= \frac{\prod_{i \in J} (1 - \pi_i) \cdot \prod_{i \in [n] \setminus J} \pi_i}{|\bigcap_{i \in J} R_s^1(i) \cap \bigcap_{i \in [n] \setminus J} R_s^2(i)|}
\end{aligned}$$

$$\begin{aligned}
&= \frac{|\bigcap_{i \in J} R_s^1(i) \cap \bigcap_{i \in [n] \setminus J} R_s^2(i)|}{|R_s|} \cdot \frac{1}{|\bigcap_{i \in J} R_s^1(i) \cap \bigcap_{i \in [n] \setminus J} R_s^2(i)|} \\
&= \frac{1}{|R_s|},
\end{aligned}$$

which is equivalent to the probability under the basic decision rule. ■

Note, that a simultaneous parametrization of binary questions can be used to model a non-binary question.

In Appendix C, the basic decision rule used for the rally-SSO-MDP is specified together with a parametrization of the decision questions presented above. Looking at the reasonable action sets and the parameterizations in Appendix C, it is observed that each binary question splits the set of reasonable actions in two equally large sets. Also, in serving states, where two binary questions simultaneously apply, each intersection of subsets is equally large. In this particular case, it is possible to set π_i to $\frac{1}{2}$ for all binary questions i and the basic decision rule is regained.

Observation 4.3.6:

Assume the setting of Proposition 4.3.5 with the additional property that

$$\left| \bigcap_{i \in J} R_s^1(i) \cap \bigcap_{i \in [n] \setminus J} R_s^2(i) \right| = \left| \bigcap_{i \in J'} R_s^1(i) \cap \bigcap_{i \in [n] \setminus J'} R_s^2(i) \right|, \forall J, J' \subseteq [n].$$

Then, setting $\pi_i = \frac{1}{2}$ for all $i \in [n]$ regains the basic decision rule.

PROOF. Since $R_s^1(i)$ and $R_s^2(i)$ for each i is a partition of R_s , the intersections $\bigcap_{i \in J} R_s^1(i) \cap \bigcap_{i \in [n] \setminus J} R_s^2(i)$ of all $J \subseteq [n]$ also form a partition of R_s . Since all intersections are equally large, they must contain exactly $(\frac{1}{2})^n \cdot |R_s|$ elements. By setting $\pi_i = \frac{1}{2}$ for all $i \in [n]$, we get

$$\prod_{i \in J} (1 - \pi_i) \cdot \prod_{i \in [n] \setminus J} \pi_i = \left(\frac{1}{2}\right)^n = \left(\frac{1}{2}\right)^n \cdot \frac{|R_s|}{|R_s|} = \frac{|\bigcap_{i \in J} R_s^1(i) \cap \bigcap_{i \in [n] \setminus J} R_s^2(i)|}{|R_s|}$$

and by Proposition 4.3.5 the resulting probability distribution for this parameter values regains the basic decision rule. ■

In the field attack situation, where also two binary questions simultaneously apply, the intersections of subsets are not equally large. This can be seen in the implementation of the basic decision rule in Listing C.14. The possible target fields of a planned shot include – in contrast to the smash – the fields directly at the net. Therefore, the number of reasonable border/non-border fields for a planned shot differs from the number of reasonable border/non-border fields of a smash. However, the parameter $\pi_{b,tech}^{field}(P_1)$ can be set to 0.4 and $\pi_{b,field}^{field}(P_1)$ to 0.5 such that the requirement of Proposition 4.3.5 is satisfied. So, for the example questions, the basic decision rule equals:

parameter	value	Describes probability that ...
π_b	0.5	... player 1 is blocking.
$\pi_{b,tech}^{serve}(P_1)$	0.5	... player 1 makes a jump serve.
$\pi_{b,field}^{serve}(P_1)$	0.5	... player 1 aims with his serve into a border field.
$\pi_{b,tech}^{field}(P_1)$	0.4	... player 1 makes a smash.
$\pi_{b,field}^{field}(P_1)$	0.5	... player 1 aims with his attack into a border field.
$\pi_{b,tech}^{serve}(P_2)$	0.5	... player 2 makes a jump serve.
$\pi_{b,field}^{serve}(P_2)$	0.5	... player 2 aims with his serve into a border field.
$\pi_{b,tech}^{field}(P_2)$	0.4	... player 2 makes a smash.
$\pi_{b,field}^{field}(P_2)$	0.5	... player 2 aims with his attack into a border field.
π_s	0.5	... it is served on opponent player 1.

Table 4.8: Parameters settings for regaining basic decision rule

4.3.3 Application to a Match

The rally-SSO-MDP should be applied to the beach volleyball tournament of the Olympic games 2012 in London. Analogously to the set-SSO-MDP the goal is to give the German team a strategic recommendation for the final match.

The input data for the rally-SSO-MDP are the individual player skills described in Subsection 4.3.1. Since the tournament lies in the past, only existing video material from the tournament can be used to estimate the player skills of both teams. In general, this is not the preferred way to determine the players' skills. The advantage of individual player skills is that they only depend on an individual player and not on other players participating in a match. So, it is possible to estimate them from individual training sessions. However, in the given situation, where no direct contact to the teams exists, it is at least also possible to use all videos of the pre-final matches of both teams to estimate the individual player skills.

Data Collection

All videos of matches of Brink-Reckermann have been evaluated to give a strategic recommendation for the German team Brink-Reckermann regarding the Olympic final. The opponent team in the Olympic final were the Brazilians Alison and Emanuel. The matches of the Brazilian players were evaluated too to get an estimate of their skills. In total six matches for each team plus the Olympic final have been analyzed.

The data was extracted from publicly available video material of the Olympics channel⁷ on YouTube. The videos cover almost all matches completely. Only five to ten rallies were not covered or not recognizable on the video material available on YouTube.

⁷<https://www.youtube.com/user/olympic>

A special software, called *Beach Volleyball Tracker*, was developed by Ronan Richter, a student assistant, for an easier annotation and data processing. The *Beach Volleyball Tracker* is a JavaScript application that runs in a browser. Figure 4.7 presents the interface used for the annotation of the videos. On the left half of the screen, the user can load and watch a video. There are different options for controlling the playback speed. On the right half of the screen, a sketch of the court is presented. The user can record the positions of the players or the ball by clicking on the corresponding points of the court sketch. In the lower half, there are options to classify hits according to the rally-SSO-MDP. Finally, the outcome of the hit or block can be specified. With some experience and the use of short-cuts, around 1.5 times the real match time is needed to analyze one match for one of the teams. The annotation of the 13 videos (6 pre-final matches for each team plus the final match) was done in several iterations and with the support of Ronan Richter and Fabian Buck, who is a sports student with practical experience in beach volleyball.

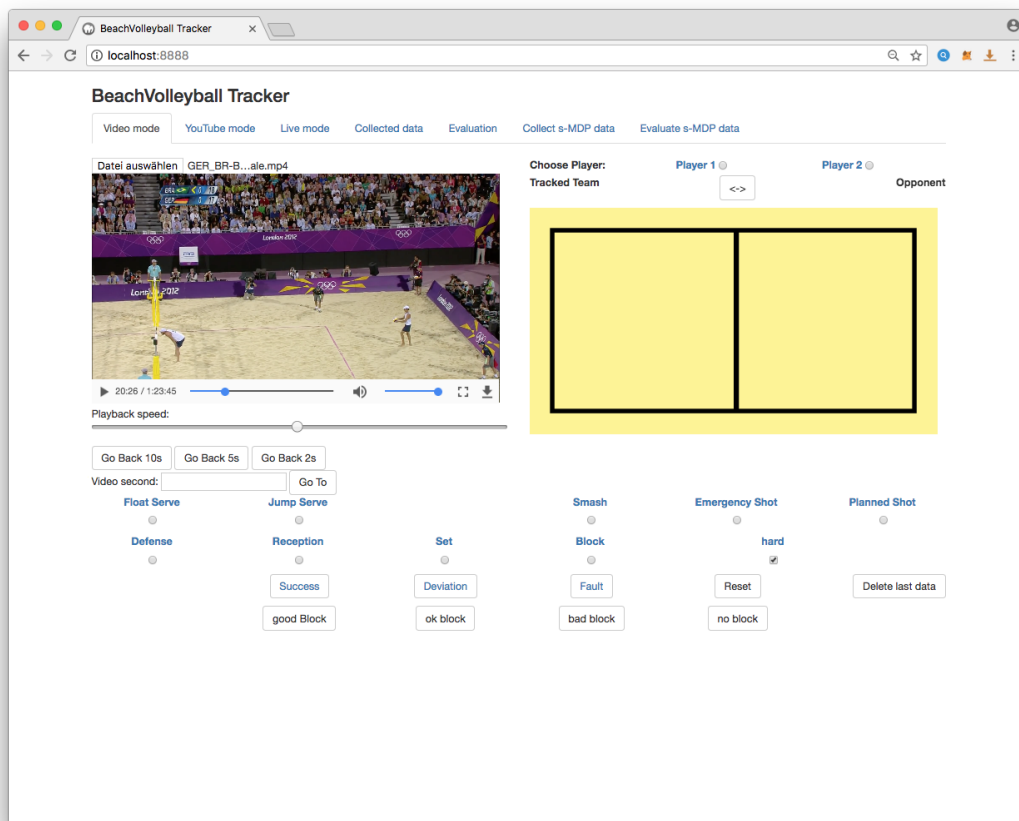


Figure 4.7: Beach Volleyball Tracker – User Interface

For estimating the individual player skills, each execution of a hitting technique *tech* has been classified according to the rally-SSO-MDP. An estimate of the blocking skills is required for the transition function of the rally-SSO-MDP. Therefore, also all blocks were annotated. By definition, a block is a

move and no hit and has other possible outcomes than a hit.

For a single event, which may be a hit, a hitting attempt or a block, the following data is saved

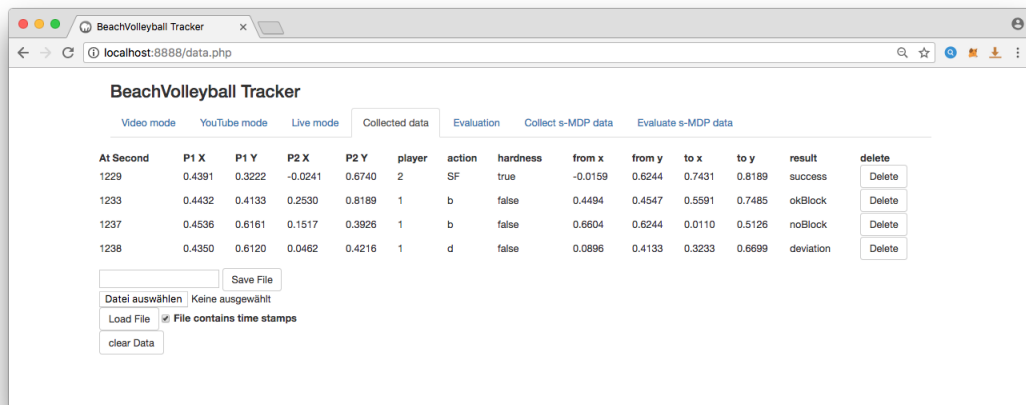
time stamp, $pos_{(x,y)}(P_1)$, $pos_{(x,y)}(P_2)$, *hitting player*, *tech*, *hardness*, $pos_{(x,y)}^t(ball)$, $pos_{(x,y)}^{t+1}(ball)$, ω .

The first value is a time stamp containing the second of the video at which the event occurred. The function $pos_{(x,y)} \in [-0.5, 1.5]$ describes the position of an object (player or ball) on the court regarding relative coordinates. The upper left corner has the relative coordinates (0, 0) and the lower right corner the coordinates (1, 1). Since the ball may also be outside the border lines, there are also values less than 0 and greater than 1 possible. At this stage, coordinates and no field names were used to be able to change the definition of the court grid. If the court grid changed for some reason, the collected data could be evaluated according to the new court grid. The *Beach Volleyball Tracker* is designed for evaluating one team at a time. Therefore a whole data stream, like in Figure 4.8, is collected for one team only. Assume the data stream belongs to a team P . In one observation, the position of the players P_1 and P_2 of the observed team is saved, and by *hitting player* $\in \{1, 2\}$, it is denoted which of the two players performed the hit. The hitting technique or block is denoted by *tech* $\in \{SJ, SF, FS, FE, FP, r, r_m, s, d, d_m, b\}$. The variable *hardness* indicates whether the ball was played *hard* or *normal* when arriving at $pos_{(x,y)}^t(ball)$. The information about the *hardness* of the ball is important for receiving and defending actions. In the rally-SSO-MDP, different receiving [defending] skills are used for modeling the reception [defense] of *hard* and *normal* balls. Finally, the resulting position $pos_{(x,y)}^{t+1}(ball)$ of the ball is saved as well as the outcome ω of the hit or the block. In a training session, it would be possible to obtain information about the player's target position. In a video recording of a real match, we do not have that information. However, we inferred that no player would on purpose hit the ball into an outside field. So, every hit that landed in an outside field has been marked with the outcome $\omega = dev$. All other hits that are no *fault* have been marked as *succ*. Through this simplification, the deviation rate is lower as if the target fields were known, while the success rate is higher. However, the system dynamics of the simulation tool presented later have been adapted to that issue. The outcome of a block is different from a hit and either a *block-point*, *block-fault*, *block-ok* or *noblock*.

Figure 4.8 shows a small data example that was collected with the Beach Volleyball Tracker. All collected data was saved in a text file. This text file is called the raw data file of the rally-SSO-MDP and has the extension `.gdata` in reference to the name the rally-SSO-MDP has in the Two-Scale approach. In total, this means, we collected 1857 events of the Brazilians Alison and Emanuel and 1635 events of the Germans Brink and Reckermann including the Olympic final. In the following the raw data of the rally-SSO-MDP will be called g-data.

Validity of Raw Data

A comparison with available statistics on the Internet is made to check the validity of the collected g-data. Two sources that contain statistics of the Olympic beach volleyball tournament in London have been found. One source is from the Fédération Internationale de Volleyball (FIVB) and contains player- as well as team-statistics of the tournament (Fédération Internationale De Volleyball, 2012b; Fédération Internationale De Volleyball, 2012a). The other source (Database, 2018) is called Beach Volleyball Database and includes besides player- and team-statistics also match wise statistics. None of the found data sources contained positional data. However, parts of the statistics can be compared to aggregated values of the data collection for the rally-SSO-MDP.



The screenshot shows a web browser window titled "BeachVolleyball Tracker" with the URL "localhost:8888/data.php". The application has several tabs: "Video mode", "YouTube mode", "Live mode", "Collected data" (selected), "Evaluation", "Collect s-MDP data", and "Evaluate s-MDP data". Below the tabs is a table with 14 columns: "At Second", "P1 X", "P1 Y", "P2 X", "P2 Y", "player", "action", "hardness", "from x", "from y", "to x", "to y", "result", and "delete". There are four rows of data. Below the table are controls for saving and loading files, including a "Save File" button, a "Datei auswählen" button, a "Load File" button, a checked checkbox for "File contains time stamps", and a "clear Data" button.

At Second	P1 X	P1 Y	P2 X	P2 Y	player	action	hardness	from x	from y	to x	to y	result	delete
1229	0.4391	0.3222	-0.0241	0.6740	2	SF	true	-0.0159	0.6244	0.7431	0.8189	success	Delete
1233	0.4432	0.4133	0.2530	0.8189	1	b	false	0.4494	0.4547	0.5591	0.7485	okBlock	Delete
1237	0.4536	0.6161	0.1517	0.3926	1	b	false	0.6604	0.6244	0.0110	0.5126	noBlock	Delete
1238	0.4350	0.6120	0.0462	0.4216	1	d	false	0.0896	0.4133	0.3233	0.6699	deviation	Delete

Figure 4.8: Beach Volleyball Tracker – Example Data

Although both sources are slightly different, all information contained in both sources is completely identical. Table 4.9 gives a summary of the information extracted from the Internet statistics that can be related to data collected for the rally-SSO-MDP.

Player	attacks	kills	errors	services	aces	service errors
Brink	163	81	0	156	10	16
Reckermann	177	104	0	151	13	23
Alison	123	73	0	170	8	27
Emanuel	254	128	0	164	3	14

Table 4.9: Internet statistics of the Olympic Beach Volleyball Tournament 2012 in London – Final match included

The corresponding aggregated values of the g-data events were calculated to compare the collected g-data with the presented statistics from the Internet sources. By using regular expressions as search strings, this can be done without much overhead. All used search strings aggregate over the different positions and hitting techniques that belong to a particular class of hits. For example, to compare the total number of field attacks, it has been searched for all events where a *FS*, *FP* or *FE* was used by a specified player from any position on the court. Tables 4.10 and 4.11 present the used search strings and the corresponding number of occurrences in the raw data file regarding field attacks and services.

The collected g-data contains two more attacks (342) of the German team than the Internet statistics (five more attacks of Brink and three attacks less of Reckermann). For the Brazilian team, we observed in total eight attacks more. The maximum deviation between the Internet statistics and the collected g-data is for a single player less than 3.5%.

The number of successful attacks in the collected g-data is of course much higher than the number of kills denoted in the Internet statistics. A successful attack in terms of the rally-SSO-MDP means the

observed properties	$\varrho \in \{1, 2\}, tech \in \{F_{SM}, F_P, F_S\}$			
		$\omega = succ$	$\omega = fault$	
Search string	ϱ	$;\varrho;F.*$	$;\varrho;F.*;succ$	$;\varrho;F.*;fault$
Brink	1	168	150	6
Reckermann	2	174	162	5
Alison	1	127	114	6
Emanuel	2	258	229	12

Table 4.10: Attack Statistic from collected g-data – final match included

hit could be performed successfully and the ball landed in the targeted field. It does, in contrast to a kill, not include that the opponent team was not able to defend the attack and a point was made. So, the high number of successful attacks is no contradiction to the small number of kills, and those numbers should not be compared as they describe different things.

However, the number of attack faults should correspond to the number of errors. An attack fault in terms of the rally-SSO-MDP means that an execution fault occurred and the ball did not cross the net. According to the Internet statistics, none of the finalists has made an attack error in the whole tournament. An example of an attack fault proves that zero attack errors cannot be correct: An attack error of the Brazilian team can be found in the match Brazil versus Lativa in the second set. At a standing of 10 to 9 for Brazil, the Latvian team is serving, Emanuel receives the ball and Alison attacks by a smash. Alison's smash goes right into the net and Brazil loses the point. The reader may watch this sequence on the YouTube video <https://www.youtube.com/watch?v=vKgm9jg2m6c> at the 31st minute.⁸

observed properties	$\varrho \in \{1, 2\}, tech \in \{S_J, S_F\}$				
		$\omega = succ$	$\omega = fault$	$\omega \in \{fault, dev\}$	
Search string	ϱ	$;\varrho;S.*$	$;\varrho;S.*;succ$	$;\varrho;S.*;fault$	$;\varrho;S.*;(fault dev)$
Brink	1	153	137	8	16
Reckermann	2	150	127	10	23
Alison	1	170	143	9	27
Emanuel	2	164	149	10	15

Table 4.11: Serve Statistic from collected g-data – final match included

The total number of serves listed in the g-data is of the same magnitude as stated in the Internet statistics. For the Brazilians, the number of serves from the Internet statistics is for each player even the same as the number of observed serves listed in the g-data. The maximum deviation for the number of serves of the German players is less than 2%. This deviation for the number of serves of the German players can probably be explained by the fact that there four serves of the German team missing in the

⁸As mentioned before, the information is got by an e-mail conversation with the admin of the www.bvbinfo.com webpage that the number of attack errors was not available for that tournament.

video sequences.

As in the attack case, the number of successful serves is per definition much larger than the number of aces. A successful serve in terms of the rally-SSO-MDP means that the serve was successfully performed and the ball landed in the targeted field. The serve may or may not be successfully received by the opponent team. So, the number of successful serves includes the number of aces and many more serves.

The number of service errors of the Internet statistic is up to twice as large as the number of executions faults in the data. The reason for this difference is that a point loss may also occur if the ball crossed the net but deviated into an outside field. So in general, it must hold that the number of serves with an outcome $\omega \in \{succ, dev\}$ is greater or equal than the number of serving errors. In the particular case of the data collection from video sequences, a deviation was only inferred if the ball landed in an outside field. So in this case, the number of serving errors should be equal to the number of observed execution faults plus the number of deviations of all serving techniques. If these values are compared, it can be seen that the values are nearly equal. Only for Emanuel, we observed one service error or deviation more than counted in the Internet statistics.

Summarizing this comparison up: There are differences between the collected data and the data sources available on the Internet. Except for the attack errors, all the differences are relatively small in comparison to the total number of collected events. For the case of the attack faults, it is known that there is misinformation in the Internet statistics.

A further check of the collected data was done by comparing them to the collected s-data. For this purpose, sequences of collected hits in the data were composed to field attack complexes (subsequences of reception/defence-set-attack hit). The result is a sequence of services and field attacks. This composing of hits to field attack complexes is done in the script `evaluate-sdata.js` that can be found in the supplementary material of this thesis, see Appendix E. The sequence of services and field attacks has been compared to the collected s-data of the match. Note, that the g-data file contains only hits of one team. The constructed sequence of services and field attacks can be compared to the s-data of one team. Thereby, it has been ensured that both data collections are in perfect synchronization. Of course, at the beginning, the s-data was not perfectly aligned to the g-data. However, when a difference was detected by using the `evaluate-sdata.js`-script, the corresponding rally was watched again and depending on the situation either the s-data or g-data was changed to the underlying situation. The resulting sequence of services and field attacks composed from the g-data can be found in the files with the ending `.sprobs`.

Evaluation of Raw Data

As a first step, the $pos_{(x,y)}$ coordinates must be evaluated to determine the corresponding *field* of the court grid. For this purpose, the JavaScript file `evaluate-gdata.js` is used. The court grid, presented in Figure 4.6, is translated to the following vertical and horizontal lines in terms of the relative coordinates:

vertical lines:	0	0.21875	0.46875	0.5	0.53125	0.78125	1
horizontal lines:	0	0.125	0.5	0.875	1		

A JavaScript function finds the lines between which the $pos_{(x,y)}$ coordinates lie, from which in turn the field description can be determined. However, even if seven matches are regarded, not all combinations of every player positions and target fields appear in the collected data. This is due to the large number of required input probabilities:

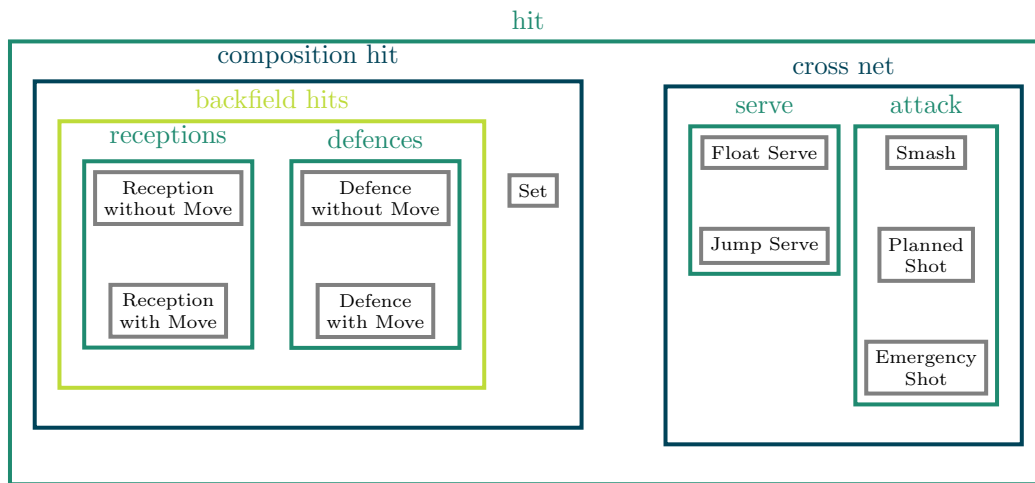


Figure 4.9: Aggregation Scheme

For example, a serve can be played from four different player positions ($P01 - P04$) to eight different target fields ($Q11 - Q24$). Together with two different hitting techniques, there exist 64 different service types for one player. Roughly half of the 1635 events of the Brink-Reckermann team belong to one player. So, in total around 800 observations per player have been collected. It is likely that there are combinations with only a few or even no observations.

To overcome this problem at least partially, an aggregation of fields to field categories is made. A field category is defined as a set of fields that has the same distance to the net. For example, $Q21 - Q24$ is one field category. It is assumed that a player may successfully hit the ball in one of the fields of a category with the same probability. The absolute number of observations from this aggregation is stated in the #-column of the Tables 4.12 to 4.18. In Appendix A, the same g-data is presented for each player with the difference that the observations are split up into events observed in pre-final matches and events from the final match only.

Especially for the attack hits, there are still some cases with very small or no observed events. An approach to handle this is to collect more g-data events from matches not belonging to the Olympic tournament. However, there may still be some cases with no observed events because the player does not perform this hit due to strategic considerations or individual preferences. As mentioned before, the best would be to get in touch with the teams and design individual training sessions to collect the players' skills. However, in the given situation, a further aggregation is made to get a reasonable amount of observations per specified hit.

In a first step, the aggregation scheme, which is presented in Figure 4.9, has been defined. The ten hitting techniques of the rally-SSO-MDP are described in the inner gray boxes. Similar techniques are grouped into categories, which are again grouped into larger categories until they become the most general category, which is just a general *hit*. For each specified hit and each category of hits, the individual success [or fault] probability can be estimated by dividing the number of desired events through the total number of events in that category. This estimation is a common maximum likelihood estimation

(Mitchell, 2017). Let $p_{succ,g}(tech_{target})$, respectively $p_{fault,g}(tech_{target})$, be that maximum likelihood estimation for a hit of player g . Assume, the reasonable number $k \in \mathbb{N}$ of observations that is needed as a minimum to estimate an individual success probability $p_{succ,g}$ is known. Then, for a hit $tech_{target}$ with $n < k$ observations the aggregation works as follows:

Definition 4.3.7 (Aggregated Skill Estimates).

Let $k \in \mathbb{N}$ be the minimum number of observations needed and $tech_{target}$ be a hit with $n < k$ observations for a player g . Let cat be the least aggregated category that contains at least k observations and includes the observations of the specified hit $tech_{target}$. The notation $p_{type,g}(cat)$, $type \in \{succ, fault, dev\}$ is used for the maximum likelihood estimate using the number of events in the category cat .

Then, the aggregated skill estimates are defined as

$$\bar{p}_{type,g}(tech_{target}) := \frac{n \cdot p_{type,g}(tech_{target}) + (k - n) \cdot p_{type,g}(cat)}{k}, \quad type \in \{succ, fault, dev\}.$$

So, the probability of the specified hit is filled up to k observations with the estimated probability of the next category that contains at least k events. Finally, the question arises how the minimum number k for a reasonable estimation is determined. Missing a general concept for this⁹, several k values were tested in a simulation of the rally-SSO-MDP for the data set. The value $k = 11$ was determined as the smallest k for which the results of a rally-SSO-MDP simulation stopped making large jumps.

In Tables 4.12 to 4.18, the skill estimations based on the described aggregation procedure with $k = 11$ are presented. In round brackets, the original maximum likelihood estimation before the aggregation is denoted. The reader may keep in mind that the presented probabilities do not need to sum up to 1 since the outcome of a hit may be a success, a fault or a deviation. So, the remaining probability is the probability of a deviation. If the number of observations is zero, it is denoted with “-” that there did not exist a maximum likelihood estimate before the aggregation. Remember that in the rally-SSO-MDP serves, smashes and emergency shots may not be played in a target field directly behind the net. The observations collected in the g-data confirmed this assumption. So the $Q31 - Q34$ [$P31 - P34$] column is empty for services, smashes and emergency shots.

⁹Krause developed a concept to derive the optimal amount of representative information by optimizing between estimator convergence and heterogeneity of the data. However, for applying this concept, a distance function would be needed. Quantification of the similarity of different hits shifts the problem to a new estimation issue.

Brink

target fields		Q11-Q14				Q21-Q24				Q31-Q34			
performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	
Serve													
S_F	Po1 - Po4	45	0.91 (0.91)	0.00 (0.00)	53	0.91 (0.91)	0.09 (0.09)	-	-	-	-	-	
S_J		38	0.92 (0.92)	0.00 (0.00)	17	0.76 (0.76)	0.18 (0.18)	-	-	-	-	-	
Attack-Hit													
FS	out	0	0.88 (-)	0.02 (-)	0	0.88 (-)	0.02 (-)	-	-	-	-	-	
	P11-P14	0	0.88 (-)	0.02 (-)	0	0.88 (-)	0.02 (-)	-	-	-	-	-	
	P21-P24	65	0.88 (0.88)	0.03 (0.03)	21	0.90 (0.90)	0.00 (0.00)	-	-	-	-	-	
	P31-P34	9	0.80 (0.78)	0.00 (0.00)	3	0.91 (1.00)	0.01 (0.00)	-	-	-	-	-	
FE	out	0	0.71 (-)	0.10 (-)	1	0.74 (1.00)	0.09 (0.00)	-	-	-	-	-	
	P11-P14	0	0.71 (-)	0.10 (-)	1	0.74 (1.00)	0.09 (0.00)	-	-	-	-	-	
	P21-P24	8	0.65 (0.63)	0.12 (0.13)	9	0.77 (0.78)	0.11 (0.11)	-	-	-	-	-	
	P31-P34	1	0.65 (0.00)	0.09 (0.00)	1	0.74 (1.00)	0.09 (0.00)	-	-	-	-	-	
FP	out	0	0.96 (-)	0.04 (-)	0	0.96 (-)	0.04 (-)	0	0.96 (-)	0.04 (-)	0	0.96 (-)	0.04 (-)
	P11-P14	0	0.96 (-)	0.04 (-)	0	0.96 (-)	0.04 (-)	0	0.96 (-)	0.04 (-)	0	0.96 (-)	0.04 (-)
	P21-P24	9	0.99 (1.00)	0.01 (0.00)	35	0.97 (0.97)	0.03 (0.03)	0	0.96 (-)	0.04 (-)	0	0.96 (-)	0.04 (-)
	P31-P34	2	0.97 (1.00)	0.03 (0.00)	3	0.88 (0.67)	0.12 (0.33)	0	0.96 (-)	0.04 (-)	0	0.96 (-)	0.04 (-)

Table 4.12: Input data from all matches: Julius Brink – Serves and Attack-Hits

attack strength		<i>normal</i>				<i>hard</i>		
performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>		
Defense	d	24	0.83 (0.83)	0.08 (0.08)	23	0.57 (0.57)	0.35 (0.35)	
	d_m	45	0.69 (0.69)	0.27 (0.27)	29	0.34 (0.34)	0.59 (0.59)	
Reception	r	35	0.97 (0.97)	0.03 (0.03)	10	0.81 (0.80)	0.09 (0.10)	
	r_m	53	0.94 (0.94)	0.02 (0.02)	6	0.98 (1.00)	0.01 (0.00)	
Set	s	157	0.99 (0.99)	0.00 (0.00)	-	-	-	
performance	#	<i>block-point</i>	<i>block-ok</i>	<i>block-fault</i>	<i>noblock</i>			
Block	b	6	0.17	0.17	0.17	0.50		

Table 4.13: Input data from all matches: Julius Brink – Defense, Reception, Set, Block

Reckermann

target fields		Q11-Q14			Q21-Q24			Q31-Q34		
performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	
Serve										
S_F	P01 - P04	39	0.82 (0.82)	0.00 (0.00)	43	0.93 (0.93)	0.05 (0.05)	-	-	-
S_J		41	0.83 (0.83)	0.02 (0.02)	27	0.74 (0.74)	0.26 (0.26)	-	-	-
Attack-Hit										
FS	out	0	0.95 (-)	0.02 (-)	0	0.95 (-)	0.02 (-)	-	-	-
	P11-P14	0	0.95 (-)	0.02 (-)	0	0.95 (-)	0.02 (-)	-	-	-
	P21-P24	68	0.97 (0.97)	0.00 (0.00)	33	0.88 (0.88)	0.06 (0.06)	-	-	-
	P31-P34	9	0.99 (1.00)	0.00 (0.00)	3	0.96 (1.00)	0.01 (0.00)	-	-	-
FE	out	0	0.93 (-)	0.00 (-)	1	0.94 (1.00)	0.00 (0.00)	-	-	-
	P11-P14	0	0.93 (-)	0.00 (-)	0	0.93 (-)	0.00 (-)	-	-	-
	P21-P24	7	0.88 (0.86)	0.00 (0.00)	5	0.96 (1.00)	0.00 (0.00)	-	-	-
	P31-P34	0	0.93 (-)	0.00 (-)	1	0.94 (1.00)	0.00 (0.00)	-	-	-
FP	out	0	0.89 (-)	0.04 (-)	0	0.89 (-)	0.04 (-)	0	0.89 (-)	0.04 (-)
	P11-P14	0	0.89 (-)	0.04 (-)	0	0.89 (-)	0.04 (-)	0	0.89 (-)	0.04 (-)
	P21-P24	4	0.75 (0.50)	0.03 (0.00)	33	0.94 (0.94)	0.03 (0.03)	1	0.90 (1.00)	0.04 (0.00)
	P31-P34	0	0.89 (-)	0.04 (-)	8	0.88 (0.88)	0.10 (0.13)	0	0.89 (-)	0.04 (-)

Table 4.14: Input data from all matches: Jonas Reckermann – Serves and Attack-Hits

attack strength		<i>normal</i>			<i>hard</i>		
performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	
Defense	d	28	0.86 (0.86)	0.07 (0.07)	2	0.77 (0.50)	0.05 (0.00)
	d_m	25	0.84 (0.84)	0.08 (0.08)	1	0.73 (0.00)	0.20 (1.00)
Reception	r	34	1.00 (1.00)	0.00 (0.00)	12	0.83 (0.83)	0.08 (0.08)
	r_m	75	0.95 (0.95)	0.03 (0.03)	10	0.81 (0.80)	0.09 (0.10)
Set	s	152	0.97 (0.97)	0.01 (0.01)	-	-	-
performance	#	<i>block-point</i>	<i>block-ok</i>	<i>block-fault</i>	<i>noblock</i>		
Block	b	263	0.11	0.12	0.14	0.63	

Table 4.15: Input data from all matches: Jonas Reckermann – Defense, Reception, Set

Alison

target fields performance		#	P ₁₁ -P ₁₄		#	P ₂₁ -P ₂₄		#	P ₃₁ -P ₃₄	
			<i>succ</i>	<i>fault</i>		<i>succ</i>	<i>fault</i>		<i>succ</i>	<i>fault</i>
Serve										
S_F		51	0.86 (0.86)	0.00 (0.00)	48	0.96 (0.96)	0.04 (0.04)	-	-	-
S_J	Q ₀₁ -Q ₀₄	52	0.73 (0.73)	0.06 (0.06)	19	0.79 (0.79)	0.21 (0.21)	-	-	-
Attack-Hit										
FS	out	0	0.88 (-)	0.06 (-)	0	0.88 (-)	0.06 (-)	-	-	-
	Q ₁₁ -Q ₁₄	0	0.88 (-)	0.06 (-)	0	0.88 (-)	0.06 (-)	-	-	-
	Q ₂₁ -Q ₂₄	56	0.91 (0.91)	0.04 (0.04)	24	0.83 (0.83)	0.08 (0.08)	-	-	-
	Q ₃₁ -Q ₃₄	9	0.98 (1.00)	0.01 (0.00)	6	0.77 (0.67)	0.21 (0.33)	-	-	-
FE	out	0	0.92 (-)	0.00 (-)	0	0.92 (-)	0.00 (-)	-	-	-
	Q ₁₁ -Q ₁₄	1	0.93 (1.00)	0.00 (0.00)	1	0.93 (1.00)	0.00 (0.00)	-	-	-
	Q ₂₁ -Q ₂₄	5	0.87 (0.80)	0.00 (0.00)	5	0.96 (1.00)	0.00 (0.00)	-	-	-
	Q ₃₁ -Q ₃₄	0	0.92 (-)	0.00 (-)	1	0.93 (1.00)	0.00 (0.00)	-	-	-
FP	out	0	0.95 (-)	0.00 (-)	0	0.95 (-)	0.00 (-)	0	0.95 (-)	0.00 (-)
	Q ₁₁ -Q ₁₄	0	0.95 (-)	0.00 (-)	0	0.95 (-)	0.00 (-)	0	0.95 (-)	0.00 (-)
	Q ₂₁ -Q ₂₄	2	0.87 (0.50)	0.00 (0.00)	10	1.00 (1.00)	0.00 (0.00)	0	0.95 (-)	0.00 (-)
	Q ₃₁ -Q ₃₄	2	0.96 (1.00)	0.00 (0.00)	5	0.97 (1.00)	0.00 (0.00)	0	0.95 (-)	0.00 (-)

Table 4.16: Input data from all matches: Alison Cerutti – Serves and Attack-Hits

attack strength		<i>normal</i>			<i>hard</i>		
performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	
Defense	d	36	0.78 (0.78)	0.08 (0.08)	8	0.46 (0.38)	0.21 (0.25)
	d_m	23	0.74 (0.74)	0.22 (0.22)	7	0.41 (0.29)	0.47 (0.57)
Reception	r	32	0.94 (0.94)	0.00 (0.00)	6	0.98 (1.00)	0.00 (0.00)
	r_m	41	0.98 (0.98)	0.00 (0.00)	1	0.87 (0.00)	0.11 (1.00)
Set	s	232	0.98 (0.98)	0.00 (0.00)	-	-	-
performance	#	<i>block-point</i>	<i>block-ok</i>	<i>block-fault</i>	<i>noblock</i>		
Block	b	303	0.12	0.14	0.14	0.60	

Table 4.17: Input data from all matches: Alison Cerutti – Defense, Reception, Set, Block

Emanuel

target fields performance		#	P11-P14		#	P21-P24		#	P31-P34	
			<i>succ</i>	<i>fault</i>		<i>succ</i>	<i>fault</i>		<i>succ</i>	<i>fault</i>
Serve										
S_F	Q01-Q04	52	0.96 (0.96)	0.02 (0.02)	53	0.87 (0.87)	0.11 (0.11)	-	-	-
S_J		41	0.98 (0.98)	0.00 (0.00)	17	0.76 (0.76)	0.18 (0.18)	-	-	-
Attack-Hit										
FS	out	0	0.87 (-)	0.06 (-)	0	0.87 (-)	0.06 (-)	-	-	-
	Q11-Q14	0	0.87 (-)	0.06 (-)	0	0.87 (-)	0.06 (-)	-	-	-
	Q21-Q24	104	0.90 (0.90)	0.03 (0.03)	71	0.80 (0.80)	0.10 (0.10)	-	-	-
	Q31-Q34	17	0.94 (0.94)	0.00 (0.00)	8	0.78 (0.75)	0.20 (0.25)	-	-	-
FE	out	0	1.00 (-)	0.00 (-)	0	1.00 (-)	0.00 (-)	-	-	-
	Q11-Q14	1	1.00 (1.00)	0.00 (0.00)	0	1.00 (-)	0.00 (-)	-	-	-
	Q21-Q24	14	1.00 (1.00)	0.00 (0.00)	5	1.00 (1.00)	0.00 (0.00)	-	-	-
	Q31-Q34	0	1.00 (-)	0.00 (-)	2	1.00 (1.00)	0.00 (0.00)	-	-	-
FP	out	0	0.94 (-)	0.00 (-)	0	0.94 (-)	0.00 (-)	0	0.94 (-)	0.00 (-)
	Q11-Q14	0	0.94 (-)	0.00 (-)	0	0.94 (-)	0.00 (-)	0	0.94 (-)	0.00 (-)
	Q21-Q24	5	0.97 (1.00)	0.00 (0.00)	25	0.92 (0.92)	0.00 (0.00)	1	0.95 (1.00)	0.00 (0.00)
	Q31-Q34	0	0.94 (-)	0.00 (-)	4	0.96 (1.00)	0.00 (0.00)	0	0.94 (-)	0.00 (-)

Table 4.18: Input data from all matches: Emanuel Rego – Serves and Attack-Hits

attack strength		<i>normal</i>			<i>hard</i>		
performance		#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>
Defense	d	29	0.86 (0.86)	0.07 (0.07)	37	0.24 (0.24)	0.46 (0.46)
	d_m	54	0.65 (0.65)	0.31 (0.31)	29	0.34 (0.34)	0.45 (0.45)
Reception	r	82	0.93 (0.93)	0.05 (0.05)	20	0.65 (0.65)	0.05 (0.05)
	r_m	75	0.93 (0.93)	0.01 (0.01)	3	0.95 (1.00)	0.01 (0.00)
Set	s	107	1.00 (1.00)	0.00 (0.00)	-	-	-
performance		#	<i>block-point</i>	<i>block-ok</i>	<i>block-fault</i>	<i>noblock</i>	
Block	b	13	0.08	0.38	0.08	0.46	

Table 4.19: Input data from all matches: Emanuel Rego – Defense, Reception, Set, Block

4.3.4 Solving the Rally-SSO-MDP

Unless otherwise specified, all computations in this section are made on a standard personal computer (MacBook Pro with 2.7GHz Intel Core i5, 8 GB 1867 MHz DDR3).

In general, SSO-MDPs can be solved by the linear programming formulations presented in Section 3.5. In the dual LP for SSO-MDPs formulation, there exist one constraint for each state and a variable for each state-action pair.

A straightforward approach to solve the rally-SSO-MDP is to construct the dual or the primal linear program and solve it by an LP-solver. For this purpose, a list of all states is needed. As a Java-

implementation of the rally-SSO-MDP was already available from a simulation of the rally-SSO-MDP¹⁰, a Java-program was used for constructing the linear program. The Java-implementation contains Java-classes or Enums for each feature variable of a state. By iterating over all combinations of values of feature variables, an enumeration of all states was attempted. After generating around 4 millions of states, an “out of memory overflow” error occurred. The Java Garbage Collection limit was exceeded. This error means that more than 98 % of the total time has been spent on garbage collection and less than 2 % of the heap is recovered. So, even if the application would run further, the progress would be too small. The default maximum heap size of a Java application is 1 GB. (*Java SE 6 HotSpot[tm] Virtual Machine Garbage Collection Tuning*) Certainly, there exist other programming languages which could perform better. However, most likely it is also not possible to enumerate all states of the rally-SSO-MDP in a different programming language.

As mentioned in Subsection 4.3.1, there are some states of the rally-SSO-MDP that can be excluded from the state space due to beach volleyball rules that forbid them. The following rules describe restrictions of the state space due to beach volleyball specific characteristics. The factor behind each characteristic estimates the proportion of the remaining states of the original state space.

- Iterate only over blocking players that are not from the team who will perform the next hit. (factor $\frac{3}{5}$)
- For $counter = -1$, set $lastContact = \emptyset$, $blockingPlayer = \emptyset$ and $hardness = \emptyset$. (factor $\leq 1 - \frac{1}{5}(1 - \frac{1}{25}) = \frac{101}{125}$)
- For $counter = 0$, set $lastContact = \emptyset$. (factor $\leq 1 - \frac{1}{5}(1 - \frac{1}{5}) = \frac{21}{25}$)

By using the described rules, the state space can be shrunken at maximum to $\frac{2}{5}$ of the complete state space. When comparing the number of states at which the Garbage Collection limit was exceeded the state space needs to be shrunken to $\frac{1}{1000}$ of it. So, even if one or two more characteristics of beach volleyball that exclude states can be found, probably the remaining number of states will not be small enough.

In the rally-SSO-MDP, there exists a basic decision rule that guarantees a reasonable game flow, see Section 4.3.2. In each state, the basic decision rule uniformly selects an action under all *reasonable* actions. Under the assumption that the basic decision rule excludes only actions that are dominated by other actions, it might be justifiable to determine an optimal policy that uses only reasonable actions. This idea is only beneficial if the restriction to reasonable actions shrinks the set of reached states significantly. For getting more insights, a simulation procedure is used that determines the number of distinct states that occurred in a simulation run.

The results of the simulation procedure are presented in Table 4.20. The number of new explored states relative to the increase in the number of simulated rallies decreases. The absolute number of generated states does not stop to increase. However, the number of visited states under the basic decision rule is significantly smaller than the total state space (factor 10^{-6}). Since the number of explored states under the basic decision rule seems not to reach a steady level of states, it is not clear which states cannot be reached under the basic decision rule.

¹⁰The simulation can be found in the supplementary material provided in Appendix E. It is used in the context of the two-scale approach in next chapter.

# rallies	# generated states	milli sec.	$\frac{\Delta \text{states}}{\Delta \text{rallies}}$
10	44	16	-
100	365	46	3.57
1000	1813	169	1.61
10000	7122	653	0.59
50000	18098	2567	0.27
100000	29314	2986	0.22
500000	54539	13234	0.06
1000000	77946	29198	0.05
5000000	124454	507146	0.01

Table 4.20: Generated state space by basic decision rule

These computational experiences show that algorithm that needs to enumerate all states are not suitable for solving the rally-SSO-MDP. Simulation-based approaches, like reinforcement learning, could be one way to tackle the rally-SSO-MDP. Reinforcement learning belongs to the area of machine-learning and tries to find an optimal policy in an environment. This environment of a reinforcement learning problem can, for instance, be a large MDP.

In general, simulation-based approaches must weigh up exploiting what is known to maximize immediate rewards against exploring new information that may improve future performance. There exist several sampling methods that try to handle this trade-off: For instance, Thompson sampling, also called posterior sampling, is a sampling method that can be applied to reinforcement learning (Russo et al., 2018). It outperforms other sampling methods like greedy-sampling and dithering as it explores purely understood actions and does not waste too much exploration effort.

When applying Thompson sampling to Markov decision problems, a deep exploration is necessary as actions may have delayed effects. Osband et al. describes in Osband et al. (2017) an approach that extends Thompson sampling to deep exploration in the context of reinforcement learning for MDPs. This approach is applicable for MDPs with an uncertain horizon in which all policies almost surely terminate in finite time. This assumption made by Osband et al. is satisfied by SSO-MDPs, see Theorem 3.4.5. In contrast to other existing reinforcement learning algorithms that lead to deep exploration in MDPs, the randomized value function approach of Osband et al. is computationally tractable for huge state spaces. Furthermore, the proposed method recovers a polynomial regret bound when used with linear value functions. So, it could be worth to apply a randomized least-squares value iteration algorithm to approximate an optimal policy of the rally-SSO-MDP.

Another idea for solving the rally-SSO-MDP is to use a decomposition approach and to generate variables or constraints dynamically. The advantage would be that not all constraints and variables have to be listed explicitly in advance. If such a decomposition approach would work well, depends on the structure of the rally-SSO-MDP.

The rally-SSO-MDP has become so complex and large since the focus during the modeling process lied on using estimable transition probabilities. An SSO-MDP with this objective for other types of sports will probably be even more complicated. So, even if it would be possible to solve this particular rally-SSO-MDP by some problem specific method, it is more desirable to find a solution approach that

can tackle arbitrary large SSO-MDPs for different types of sports. In the next chapter such an approach is presented and called *two scale approach*.

Chapter 5

A Two-Scale Approach

The two-scale approach (2-MDP approach) is a general procedure that uses SSO-MDPs to answer strategic questions in sports games. It combines two SSO-MDPs of different granularity to benefit from the particular advantages of each model. In the last chapter, two different SSO-MDPs for beach volleyball have been introduced. The set-SSO-MDP models a beach volleyball set on a very rough level. In contrast, the rally-SSO-MDP is a much more detailed model and captures a beach volleyball rally. Both models have advantages and disadvantages as highlighted in the corresponding sections of the last chapter. In particular, it was shown that in both models there exist weaknesses when answering a strategic question prior to a match. The 2-MDP approach is a procedure for overcoming these weaknesses and answering sport-strategic questions adequately. The introduced pair of beach volleyball SSO-MDPs is a perfect candidate for the 2-MDP approach to be presented in this chapter. However, this chapter starts with a general definition of the 2-MDP approach as this approach is also applicable to other pairs of SSO-MDPs. After the general definition, a concrete implementation of the 2-MDP approach using the beach volleyball SSO-MDPs is given and applied to the Olympic final 2012 in London. The results are compared to those of Chapter 4 from using a single SSO-MDP. Finally, some new possibilities that arise from using the 2-MDP approach are presented.

5.1 General Procedure

In Chapter 3, theoretical aspects of SSO-MDPs have been analyzed. Due to modeling decisions, SSO-MDPs that model a similar strategic question can become very different as shown in the last chapter. This section focuses on the implications of certain modeling decisions for SSO-MDPs. It starts with a formal definition of a sport-strategic question. Afterward, advantages and disadvantages of different model granularities are summarized to motivate the 2-MDP approach. Finally, the s-g-implementation and the 2-MDP approach are introduced.

5.1.1 Sport-Strategic Question

The purpose of an SSO-MDP is to answer a sport-strategic question. Sport-strategic questions have already been formulated in the last chapter in connection with the set-SSO-MDP and the rally-SSO-MDP. Nevertheless, this section starts with characterizing a sport-strategic question and contains a

formal definition.

A sport-strategic question should consider a decision that can be made by a coach or player before a match. The answer should take the opponent team participating in the match into account. Therefore, the best strategy may be different for different opponents. So, a sport-strategic question is not the investigation of a general rule or principle that applies against every opponent.

Furthermore, the decision should regard an aspect that can directly and in the short term be influenced by the team or the player. One example that is not directly and in the short term influenceable by the players is the expected goal probability from a penalty kick. The team would probably have to train penalty kicks over several weeks to significantly increase this probability. Capabilities like turning a penalty kick into a goal are regarded as skills of a team or an individual player. There exist several papers that examine the importance of different skills (Heiner, Fellingham, and Thomas, 2014; Miskin, Fellingham, and Florence, 2010) or try to determine how an improvement of a skill effects the probability of winning a match (Pfeiffer, Zhang, and Hohmann, 2010; Pfeiffer, 2005). However, a sport-strategic question is considered in the short term prior to a match, and it is assumed that the skills of all participants in the game can not be changed in that short period. A sport-strategic question regards decisions that rely on those skills and can be directly put into practice.

Next, a sport-strategic question is formalized. This chapter uses the terminology of a team sport, but all considerations also apply to individual sports like, e. g., tennis or badminton. Let team P be the team whose strategy should be optimized and whose winning probability should be maximized while team Q should be the opponent team in the match under consideration. Both teams are characterized by a general skill set that is assumed to be fixed for the upcoming game:

$$Skill(team), team \in \{P, Q\}.$$

Depending on the strategic question the skill set of a team can be characterized by different parameters. Most of the time these parameters are probabilities that describe the capabilities of a team.

It is assumed that the number of different playing variants or strategies is finite in a sports game. This assumption can be justified by the fact that there are finitely many humans participating in a sports game and their discrimination between different actions is not infinitely accurate. For instance, consider a serve in a tennis match and let the speed of the ball be the subject of the strategic question. Then, first of all, there exist a minimum velocity and a maximum velocity the player can perform. Furthermore, depending on the skills of the player, he can discriminate between several speeds in between. However, there are only finitely many characteristics between which the player can distinguish deliberately. Also for a decision considering a line-up or a substitution of a player, the number of possibilities is finite since the number of players in a team is finite. Let m be the finite number of different strategies, which are denoted by

$$Strat = \{a_1, a_2, \dots, a_m\}.$$

Furthermore, the sport-strategic question concerns a particular game. The conditions of a game influence the decision. For example, for outdoor sports games the weather conditions could play a role, or in tennis, the floor covering of a match influences the decision. All these environmental parameters that characterize the particular game are summarized in a parameter set Env .

Now, all described parameters can be summed up in a sport-strategic question:

Definition 5.1.1 (Sport-Strategic Question).

A sport-strategic question concerns a type of sports and is characterized by

$$(Env, Skill(P), Skill(Q), Strat),$$

where

- *Env* contains the environmental parameters that characterize the special match under consideration.
- *Skill(P)*, *Skill(Q)* are the skill sets of the teams participating in the match.
- *Strat* is a set of strategies that can be played by team *P*.

The sport-strategic question asks which strategy *Strat* of team *P* leads to the highest probability of *P* to win the match.

If the winning probability $\mathcal{W}inProb(\cdot)$ of each strategy in the match under considerations was known, the answer could easily be given by evaluating

$$a^* = \arg \max_{a_i \in Strat} \{\mathcal{W}inProb(a_i)\}.$$

However, a direct estimation of these winning probabilities from historical data is often not possible. As defined in the sport-strategic question there are a bunch of parameters that characterize the regarded match. For a standard maximum likelihood estimation of the strategies' winning probability $\mathcal{W}inProb(\cdot)$, matches are needed where the strategy under consideration is played and which fit to all the parameters defined in the strategic question. However, in most of the time, a team faces a certain opponent only once or twice in a certain tournament or season of a league. Furthermore, relying on matches that lie further in the past is not possible since the skill sets of the teams probably have changed. There may be new players in a team, or players' skills may have improved or deteriorated. If the estimation of winning probabilities from historical data gets hard, this is the point where sport-strategy-MDPs (SSO-MDPs) can help to evaluate the winning probabilities of the strategies.

5.1.2 Modeling Granularity

For every strategic question, there always exist more than one SSO-MDP that model the subject of interest in an adequate way. In general, an SSO-MDP replaces the black box covered by the winning probability of a strategy with a model of the game mechanism. Thereby, the game mechanism can be modeled on different levels of detail. The more detailed the game mechanism is modeled, the less is captured by a single transition probability and the more insights into the game mechanisms are required.

The winning probability $\mathcal{W}inProb(\cdot)$ of a strategy can be interpreted as an extreme variant of an SSO-MDP where the complete game is considered as a black box and captured in a single transition. Such an extreme variant of an SSO-MDP is illustrated in the upper half of Figure 5.1. The other extreme

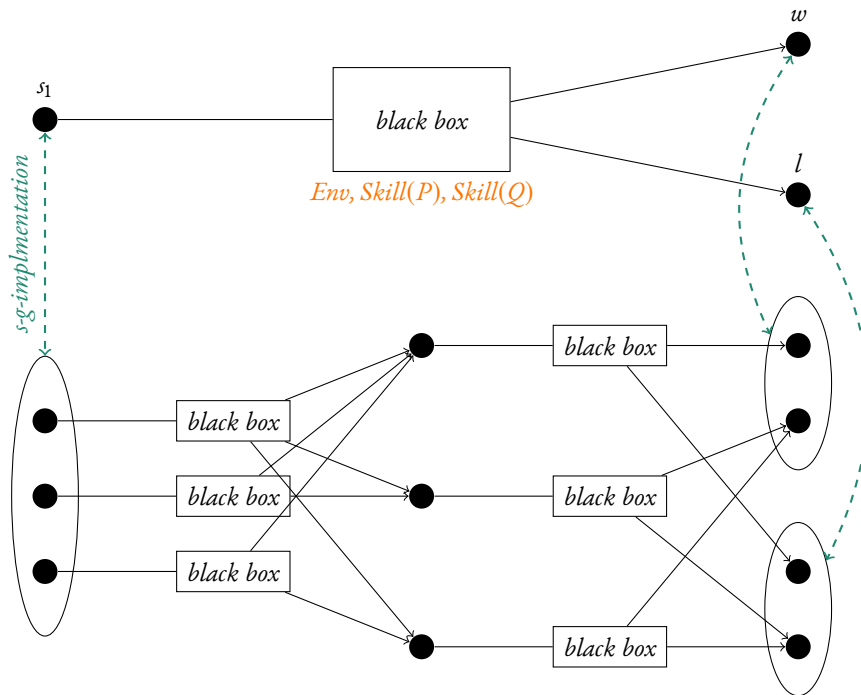


Figure 5.1: Modeling Decision – Degrees of Detail

is to model every independent random influence by a single transition. Due to the knowledge of the game mechanisms, it may be possible to identify and model those independent random influences. The resulting decomposition of a transition probability leads to the fact that a smaller subset of parameters may influence the decomposed transition. For instance, if the transition describes the outcome of a shot from a specific position by an individual soccer player, only the skills of that player and the goalkeeper instead of the whole teams may determine that transition. Of course, also a more detailed state description is necessary to be able to differentiate between transitions that only depend on a subset of the parameters of the strategic question. The lower half of Figure 5.1 illustrates an SSO-MDP with a higher level of detail for the same sport-strategic question.

As the reader may notice from the illustration of Figure 5.1, the described modeling granularity has an impact on the resulting SSO-MDP. The more detailed the game mechanism is modeled, the larger is the state and the action space. However, the transition probabilities will depend on fewer parameters, and therefore, it will be easier to find appropriate data records for estimating the transition probabilities. At the same time, of course, also a more significant number of different transition probabilities is needed. In a rough SSO-MDP, the number of states, actions, and required transition probabilities may be very small, and it may be possible to find an analytical solution for such an SSO-MDP. However, if there exists no, or not enough data that fits the parameter setting, the transition probabilities cannot be estimated, and the analytical solution cannot be evaluated for the match under consideration.

Table 5.1 summarizes the trade off between a rough and a detailed model formulation of an SSO-MDP.

	rough SSO-MDP	detailed SSO-MDP
game mechanism	black box	system dynamics
transition probabilities	depend on many parameters	depend only on a subset of parameters
size of the SSO-MDP	small state and action space	large state and action space
appropriate records	small estimation basis	larger estimation basis

Table 5.1: Comparison rough versus detailed SSO-MDP

5.1.3 The Underlying Idea

The two-scale approach tries to combine the advantages of two different granularity levels to answer strategic questions. Assume that the same strategic question is modeled by two SSO-MDPs. One SSO-MDP is very rough and considered as the strategic MDP (*s*-MDP). The other SSO-MDP is very detailed and will be called the gameplay MDP (*g*-MDP). The *g*-MDP should be a refined version of the *s*-MDP. A mapping between the states and actions of the models must be defined such that both models can be combined. This mapping is called an *s-g-implementation*. It will be formally defined in the next section. In Figure 5.1, an *s-g-implementation* of the states is illustrated.

Given an *s*-MDP, a *g*-MDP and an *s-g-implementation* regarding a strategic question, the two scale approach works as follows: The *s-g-implementation* translates the strategies *Strat* that are contained in the *s*-MDP to *g*-MDP decision rules. The transition probabilities of the *g*-MDP are estimated from appropriate data records such that the *g*-MDP can be simulated with the decision rule defined by the *s-g-implementation* for a strategy. The simulation is validated by historical data and refined if the gameplay mechanism is not correctly represented in the *g*-MDP. The result of a valid *g*-MDP simulation is an estimate for the *s*-MDP transition probabilities. Finally, the *s*-MDP is solved using the estimated transition probabilities from the *g*-MDP simulation. If an analytical solution of the *s*-MDP can be determined, it is evaluated with the estimated transition probabilities of the simulation. If no analytical solution exists, an algorithmic solution of the *s*-MDP with the estimated transition probabilities needs to be found. Since the state and action spaces of the *s*-MDP are small, a solution calculated, e.g., by the linear programming formulation of Chapter 3 should not cause many difficulties. The general procedure of the two-scale approach is illustrated in Figure 5.2.

The reader may ask himself what the benefit of the two-scale approach is compared to a single SSO-MDP. Assume, only the *g*-MDP is considered to answer a strategic question. Then, of course, a simulation can be used to determine the winning probability of a *g*-MDP-strategy. However, even if an optimal *g*-MDP-strategy is identified, this strategy would define an action choice in every state of the *g*-MDP. Due to a large number of different states in a detailed *g*-MDP, this optimal *g*-MDP-strategy could be challenging to handle and hard to remember for a player. Coming from an *s*-MDP as in the two-scale approach, there always exists an initial strategic question and a set of *s*-MDP-strategies. Together with the *s-g-implementation*, there automatically exists a practical and realizable description of the optimal *g*-MDP-strategy. Another drawback from considering only the *g*-MDP is that solving the *g*-MDP is in general harder than solving the *s*-MDP.

On the other hand, if only an MDP as rough as the *s*-MDP is recognized, an analytical solution may be found, but probably one runs into troubles when estimating the transition probabilities. As

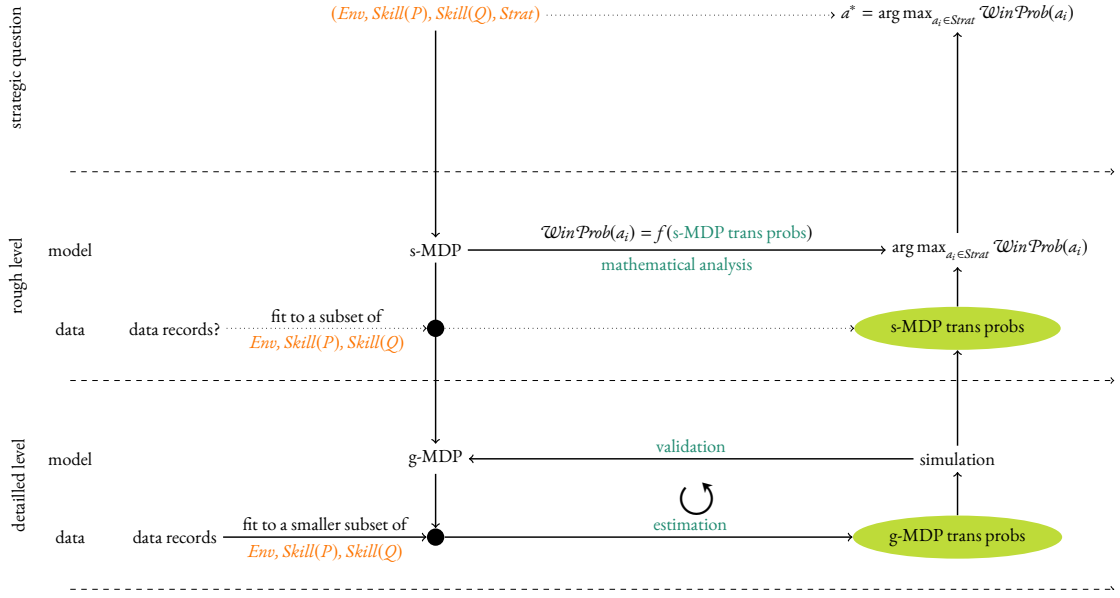


Figure 5.2: Two Scale Approach

mentioned before, it could be possible that merely no applicable data to the parameters characterizing the strategic question can be found. Furthermore, more insights about the sensitivity of an optimal strategy are possible if an analytical solution of the s-MDP can be enriched with results of the g-MDP.

After having illustrated and motivated the idea of the two-scale approach, the next section presents a formalization of the s-g-implementation.

5.1.4 Formalization

Consider two different SSO-MDPs that model the same strategic question. According to our terminology, the coarser model is called strategic MDP (s-MDP), and its parameters are denoted by

$$(S^s, \mathcal{A}^s, p(\cdot|s, a)^s, r(s, a)^s, W^s, L^s, s_1^s).$$

The second model is called the gameplay MDP (g-MDP) and its parameters are denoted by

$$(S^g, \mathcal{A}^g, p(\cdot|s, a)^g, r(s, a)^g, W^g, L^g, s_1^g).$$

According to underlying idea of the two-scale approach, the g-MDP is the more detailed model and the following relation between the state sets needs to hold

$$|S^g| > |S^s|.$$

An s-g-implementation describes the relation between the s-MDP and the g-MDP. Consider a mapping

$$\begin{aligned} \tau_S : S^s &\rightarrow 2^{S^g} \\ s^s &\mapsto \tau_S(s^s) \in S^g \end{aligned}$$

that maps an s-MDP-state to a set of g-MDP states. The idea behind this function is to map a state s^s to all states s^g of the g-MDP that are a refinement of the state s^s . Since several states in the g-MDP can be a refinement of the same s-MDP state, the codomain of the mapping is the powerset of the set of all g-MDP states. The mapping τ_S is not surjective since there may be states that have no counterpart in the s-MDP. However, it will be required that the images of different s-MDP states have no g-MDP states in common. So, no s^g is contained in the image of more than one s-MDP-state. This requirement expresses that the g-MDP is a refinement and no aggregation of the s-MDP.

Sometimes one is interested in the state $s^s \in S^s$ that is a coarsening of a g-MDP state s^g . Therefore, the inverse mapping τ_S^{-1} is defined as

$$\tau_S^{-1} : S^g \rightarrow S^s \cup \emptyset$$

$$s^g \mapsto \begin{cases} s^s & \text{if there exists an } s^s \text{ s.t. } s^g \in \tau_S(s^s), \\ \emptyset & \text{else.} \end{cases}$$

Since there may exist some g-MDP-states that are no refinement of an s-MDP-state, the codomain of τ_S^{-1} also contains an artificial element \emptyset . According to the definition of an s-g-implementation presented below, it is required that the image of τ_S satisfies the condition $\tau_S(s_1) \cap \tau_S(s_2) = \emptyset$ for each pair of s-MDP-states $s_1, s_2 \in S^s$ with $s_1 \neq s_2$. Therefore, τ_S^{-1} is always well-defined and can be derived from τ_S .

The following definition summarizes the conditions for an s-g-implementation:

Definition 5.1.2 (s-g-implementation).

Assume an s-MDP and a g-MDP that are both SSO-MDPs. The mapping $\tau_S : S^s \rightarrow 2^{S^g}$ is an s-g-implementation if

- $\tau_S(s_1) \cap \tau_S(s_2) = \emptyset$ for each pair of s-MDP-states $s_1, s_2 \in S^s$ with $s_1 \neq s_2$.
-

The definition of an s-g-implementation assures formal correctness of the mapping. However, by only adhering to that definition, meaningless implementations are still possible. The states can be mapped in a feasible manner while the available actions and transitions between state pairs could model different parts of the game mechanism. This is the reason why a second mapping τ_A that maps s-MDP actions to g-MDP decision rules is required such that the s-g-implementation becomes meaningful:

In the s-MDP, a transition probability covers a game mechanism which is to some degree explicitly modeled in the g-MDP. This replacement of a black box by a sequence of transitions must correctly reproduce the s-MDP transition probability such that the s-g-implementation is meaningful:

Definition 5.1.3 (Meaningful s-g-implementation).

An s-g-implementation $\tau_S : S^s \rightarrow 2^{S^g}$ is *meaningful* if for every action $a^s \in A^s$ and every state $s_1^s \in S^s$ there exists a g-decision rule $d^g \in D^{MR}$ such that

$$p(s_2^s | s_1^s, a^s) = \sum_{k=1}^{\infty} \mathbb{P}^{d^g} \{ X_{t+k} \in \tau_S(s_2^s) \mid X_t \in \tau_S(s_1^s), \tau_S^{-1}(X_{t+i}) = \emptyset \forall i \in \{1, \dots, k-1\} \},$$

$$\forall s_2^s \in S^s, t \in \mathbb{N},$$

where $\{X_t, t \in \mathbb{N}\}$ is the induced Markov process under decision rule d^g in the g-MDP. The mapping $\tau_A : S^s \times A^s \rightarrow D^{MR}$ stores for each state-action pairs one corresponding decision rules.

A meaningful s-g-implementation assures that for every state-action pair (s_1^s, a^s) in the s-MDP there exists a decision rule in the g-MDP that reflects the transitions of the s-MDP in the g-MDP. A transition (s_1^s, a^s, s_2^s) in the s-MDP may correspond to a sequence of transitions $(s_1^g, a_1^g, s_2^g, \dots, s_k^g)$ in the g-MDP with $s_1^g \in \tau_S(s_1^s)$ and $s_k^g \in \tau_S(s_2^s)$. However, during that sequence of transitions there may only occur g-MDP states which have no pre-image in S^s , i.e., $\tau_S^{-1}(s_i^g) = \emptyset$ for all $1 < i < k$. This condition is necessary such that a g-MDP decision rule combined from the decision rules given by the mapping τ_A can reflect an s-MDP decision rule.

Algorithm 7 describes how an s-MDP decision rule d^s can be simulated in the g-MDP by using a meaningful s-g-implementation. The simulation starts at an image $s_1^g \in \tau_S(s_1^s)$ of a predefined starting state s_1^s . The s-MDP decision rule d^s is evaluated in s_1^s to the first action choice a^s . The g-MDP decision rule d^g is then initialized by the decision rule $\tau_A(s_1^s, a^s)$. After this initialization the algorithm works as follows: As long as the current state s_i^g of the g-MDP is not a winning or a losing state, the system dynamics of the g-MDP is evaluated using the decision rule d^g . Each time the current state of the g-MDP has a pre-image in the state space S^s , the s-MDP decision rule d^s is evaluated to an action choice a^s and the g-MDP decision rule d^g gets updated by $\tau_A(\tau_S^{-1}(s_i^g), a^s)$. The output of the algorithm is a realization of g-MDP states (s_1^g, \dots, s_N^g) .

Algorithm 7: Simulate s-MDP decision rule

Data: s-MDP with a decision rule d^s and a starting state s_1^s , g-MDP, meaningful s-g-implementation with functions τ_S, τ_A
 Result: (s_1^g, \dots, s_N^g) – a realization of g-MDP states

- 1 $i \leftarrow 1$;
- 2 choose $s_i^g \in \tau_S(s_1^s)$;
- 3 evaluate $a^s \leftarrow d^s(s_1^s)$;
- 4 set $d^g \leftarrow \tau_A(s_1^s, a^s)$;
- 5 while $s_i^g \notin W^g \cup L^g$ do
 - 6 $i \leftarrow i + 1$;
 - 7 $s_i^g \leftarrow \text{evaluate g-MDP in } s_{i-1}^g \text{ using } d^g$;
 - 8 if $\tau_S^{-1}(s_i^g) \neq \emptyset$ then
 - 9 evaluate $a^s \leftarrow d^s(\tau_S^{-1}(s_i^g))$;
 - 10 set $d^g \leftarrow \tau_A(\tau_S^{-1}(s_i^g), a^s)$;
 - 11 end
- 12 end

From a realization of the g-MDP generated by Algorithm 7, estimates for the s-MDP transition probabilities under decision rule d^s , which may also be a randomized decision rule, can be made. In Section 2.1, transition probabilities of randomized decision rules are defined as

$$p(j|s, d_t(s)) = \sum_{a \in A_s} p(j|s, a) q_{d_t(s)}(a).$$

Proposition 5.1.4 describes how a maximum likelihood estimation for an s-MDP transition probability $p(s_2^s | s_1^s, d^s(s_1^s))^s$ can be made. The function $\mathbb{1}_{x \in X}$ used in Proposition 5.1.4 is 1 if $x \in X$ holds and 0 else. So, the numerator of Proposition 5.1.4 counts how often an image of state s_1^s is directly followed by an

image of s_2^s in the g-MDP realization. The denominator counts how often an image of s_1^s occurred in the g-MDP realization.

Proposition 5.1.4 (Estimate s-MDP transition probabilities form 2-MDP approach):

Let (s_1^g, \dots, s_N^g) be a realization of a g-MDP simulation according to Algorithm 7. Furthermore, let $(s_{i_1}^g, s_{i_2}^g, \dots, s_{i_{N'}}^g)$ be a subsequence of (s_1^g, \dots, s_N^g) that contains all s_i^g but only these of the realization with $\tau_S^{-1}(s_i^g) \neq \emptyset$.

Then, the transition probability from s_1^s under use of d^s to s_2^s can be estimated by

$$p(s_2^s | s_1^s, d^s(s_1^s))^s \approx \frac{\sum_{k=1}^{N'-1} \mathbb{1}_{\tau_S^{-1}(s_{i_k}^g)=s_1^s} \cdot \mathbb{1}_{\tau_S^{-1}(s_{i_{k+1}}^g)=s_2^s}}{\sum_{k=1}^{N'-1} \mathbb{1}_{\tau_S^{-1}(s_{i_k}^g)=s_1^s}}.$$

When applying the 2-MDP approach to a real match, all transition probabilities are only estimated and not entirely known. Therefore, Definition 5.1.3 may probably not be exactly met by any decision rule. However, one should keep in mind that the transition probabilities of the s-MDP as well as the transition probabilities of the g-MDP are only estimates. In the next section, a 2-MDP approach for beach volleyball is implemented and validated. The result of this validation should be interpreted together with the estimation accuracy of the transition probabilities.

A 2-MDP approach with a meaningful s-g-implementation opens up new possibilities. Assume an s-g-implementation is validated on an adequate dataset to be meaningful. If then the SSO-MDPs are applied to a new data set, where, for instance, only the g-MDP transition probabilities can be well estimated, the s-g-implementation can be used to derive analytical bounds for the s-MDP transition probabilities. The described procedure is carried out in (Hoffmeister and Rambau, 2017a) for the s-MDP approach for beach volleyball present in the next section. However, the derived bounds are not very tight due to the size of the sets in the co-domains of τ_S and τ_A . In this case, a simulation of the g-MDP is more useful to get estimates of the s-MDP transition probabilities.

5.2 A Two-Scale Approach for Beach Volleyball

This section defines a meaningful s-g-implementation for the set-SSO-MDP and the rally-SSO-MDP presented in the last chapter. In the second subsection of this section, the s-g-implementation is validated using data of the beach volleyball tournament at the Olympic games 2012 in London. In Subsection 5.2.3, the strategic question of the set-SSO-MDP is evaluated by the 2-MDP approach. These computational results are compared to the results presented in the last chapter using only the set-SSO-MDP.

At the end of this section, two new tools for coaches and players are presented that can be generated with the help of a meaningful s-g-implementation. One tool is called a *Skill-Strategy-Score-Card*, see Subsection 5.2.5, and should help coaches to decide between two alternative strategies when the skills of the own players or the performance of the opposing team vary. The other tool is a constant-sum-matrix-game, see Subsection 5.2.6, that helps to detect patterns of strategy parameters that are promisingly against a particular opponent team. Without a meaningful s-g-implementation and the 2-MDP approach, it would be hard or even impossible to generate such outputs.

5.2.1 Implementation for Beach Volleyball

After having introduced the 2-MDP approach formally, a concrete implementation for beach volleyball is specified in this subsection. It will be assumed that the reader is familiar with the definitions of the set-SSO-MDP and the rally-SSO-MDP presented in Chapter 4.

The set-SSO-MDP is the coarser of the two models and takes, therefore, the role of the strategic MDP (s-MDP) in the 2-MDP approach. In the following of this subsection the term s-MDP is used as a synonym for the set-SSO-MDP. In the same way, the rally-SSO-MDP is considered as the g-MDP in the two-scale approach implementation of this subsection.

According to the definition of the set-SSO-MDP, the state space of the s-MDP is defined as

$$\begin{aligned} S^s &= S^{\text{tie}} \cup S^{\text{reg}} \\ S^{\text{tie}} &= \{(z, k, l) \mid z \in \{-2, -1, 0, 1, 2\}, k \in \{P, Q\}, l \in \{0, 1\}\}, \\ S^{\text{reg}} &= \{(x, y, k, l) \mid x, y \in \{0, \dots, 21\} \text{ with } (x \leq 19 \vee y \leq 19), k \in \{P, Q\}, l \in \{0, 1\}\}. \end{aligned}$$

The set S^{tie} contains 20 states and the state set S^{reg} of the regular game 1920 states.

The state space of the g-MDP is huge and contains more than $5.9 \cdot 10^9$ states that are of the form

$$(pos(P_1), pos(P_2), pos(Q_1), pos(Q_2), pos(ball), counter, lastContact, hardness, blockingPlayer).$$

In Subsection 4.3.1 in Table 4.5, a categorization of the g-MDP states is defined. The categories partition the complete state set and each category is distinct such that

$$S^g = S_P^{\text{serve}} \cup S_Q^{\text{serve}} \cup S^{\text{rec}} \cup S^{\text{def}} \cup S^{\text{set}} \cup S^{\text{attack}} \cup W^g \cup L^g.$$

Each state set $S \in \{S^{\text{rec}}, S^{\text{def}}, S^{\text{set}}, S^{\text{attack}}\}$ is further differentiated in S_P respectively S_Q regarding the possession of the ball. For example, $s \in S^{\text{rec}}$ is a receiving state of team P , i.e., $s \in S_P^{\text{rec}}$ if $side(pos(ball)) = P$. The state-categories of the g-MDP are used when defining the state mapping τ_S of the s-g-implementation.

Having recapitulated the state sets of both SSO-MDPs, it has become clear that the condition $|S^g| > |S^s|$ is satisfied. For the considered SSO-MDPs define the mapping $\tau_S : S^s \rightarrow 2^{S^g}$ as

$$\begin{aligned} \tau_S(x, y, P, 1) &\mapsto S_P^{\text{serve}}, \\ \tau_S(x, y, Q, 1) &\mapsto S_Q^{\text{serve}}, \\ \tau_S(x, y, P, 0) &\mapsto S_P^{\text{set}}, \\ \tau_S(x, y, Q, 0) &\mapsto S_Q^{\text{set}} \end{aligned}$$

Since all state categories are disjoint, the mapping τ_S fulfills the condition of an s-g-implementation according to Definition 5.1.2. A field attack state of the s-MDP is mapped by τ_S to the state set of setting states in the g-MDP. This decision comes from the fact that in a reception or defending state it is not clear whether the ball can be brought under control and the team starts a field attack. The results of the validation of the s-g-implementation, see Subsection 5.2.2, support that decision.

The strategic question which is evaluated by the set-SSO-MDP in Subsection 4.2.5 asks whether a *risky* or *safe* play in a serving situation combined with a *risky* or *safe* play in the field attack situation

s-MDP			g-MDP			
$s^s \in \mathcal{S}^s$	a^s	$\tau_{\mathcal{A}}(s^s, a^s)$	π_b	$(\pi_{b,tech}^{serve}(\xi), \pi_{b,field}^{serve}(\xi), \pi_{b,tech}^{field}(\xi), \pi_{b,field}^{field}(\xi))$		π_s
				$\xi = P_1$	$\xi = P_2$	
(x, y, P, l)	<i>basic</i>	<i>basic</i>	0.5	(0.5, 0.5, 0.4, 0.5)	(0.5, 0.5, 0.4, 0.5)	0.5
(x, y, P, l)	<i>risky</i>	<i>risky</i>	0.5	(1, 1, 1, 1)	(1, 1, 1, 1)	0.5
(x, y, P, l)	<i>safe</i>	<i>safe</i>	0.5	(0, 0, 0, 0)	(0, 0, 0, 0)	0.5
(x, y, P, l)	<i>final</i>	<i>final</i>	0.02	(0.19, 0.04, 0.74, 0.26)	(0.24, 0.17, 0.69, 0.33)	0.27
(x, y, P, l)	<i>prefinal</i>	<i>prefinal</i>	0.02	(0.39, 0.19, 0.65, 0.39)	(0.50, 0.36, 0.72, 0.45)	0.5
s^s	a^s	$\tau_{\mathcal{A}}(s^s, a^s)$	π_b	$\xi = Q_1$	$\xi = Q_2$	π_s
(x, y, Q, l)	<i>basic</i>	<i>basic</i>	0.5	(0.5, 0.5, 0.4, 0.5)	(0.5, 0.5, 0.4, 0.5)	0.5
(x, y, Q, l)	<i>risky</i>	<i>risky</i>	0.5	(1, 1, 1, 1)	(1, 1, 1, 1)	0.5
(x, y, Q, l)	<i>safe</i>	<i>safe</i>	0.5	(0, 0, 0, 0)	(0, 0, 0, 0)	0.5
(x, y, Q, l)	<i>final</i>	<i>final</i>	0.96	(0.68, 0.16, 0.93, 0.43)	(0.37, 0.27, 0.81, 0.54)	0.35
(x, y, Q, l)	<i>prefinal</i>	<i>prefinal</i>	0.96	(0.37, 0.18, 0.82, 0.36)	(0.35, 0.17, 0.86, 0.31)	0.5

Table 5.2: Meaningful s-g-implementation $\tau_{\mathcal{A}}(s^s, a^s) \mapsto \pi$

has the higher winning probability. Besides these two s-MDP actions, which are sometimes also called strategies, another action, named *final*, has been investigated in Subsection 4.2.5. The *final* action represents the actually played strategy in the final. Similar, to the *final* action a *prefinal* action is now introduced that represents the played strategy in the pre-final matches. For completeness, also a basic decision rule for the s-MDP is included in the action set. So, the action set of the s-MDP is

$$\mathcal{A}_{\bar{s}}^s = \{risky, safe, final, prefinal, basic\}$$

for serving states (\bar{s} with $l = 1$) as well as for field attack states (\bar{s} with $l = 0$).

In contrast, the action sets available in a state of the rally-SSO-MDP are too large to list them explicitly. That is why the concept of a parameterized basic decision rule has been introduced in Subsection 4.3.2. Its purpose is to eliminate unrealistic decision rules. It helps to focus on the strategic questions raised in Subsection 4.3.2. The parameters of the basic decision rule for the rally-SSO-MDP are

$$\pi = \begin{pmatrix} \pi_b \\ \pi_b \\ \pi_s \end{pmatrix}, \quad \pi_b = \begin{pmatrix} \pi_b^{serve} \\ \pi_b^{field} \end{pmatrix}, \quad \pi_b^{sit} = \begin{pmatrix} \pi_{b,tech}^{sit}(\xi) \\ \pi_{b,field}^{sit}(\xi) \end{pmatrix}, \quad \begin{matrix} sit \in \{serve, field\}, \\ \xi \in \{P_1, P_2\}. \end{matrix}$$

Table 4.8 presents the parameter settings for a uniform distribution over all reasonable actions. This parameter setting of the basic decision rule will be called *basic*. In the following, the mapping $\tau_{\mathcal{A}}$ of a meaningful s-g-implementation will be defined.

Under the assumption of a meaningful s-g-implementation implementation, there must exist for every state-action pair (s^s, a^s) of the s-MDP, a decision rule d^g that reflects the transitions of the s-MDP in the g-MDP. Table 5.2 presents state-action pairs (s^s, a^s) and a corresponding g-MDP decision rule $\pi = \tau_{\mathcal{A}}(s^s, a^s)$ with its parameter settings. The parameters of the basic decision rule *basic* are those specified in Table 4.8. The *risky* action of the s-MDP is implemented in the g-MDP by always choosing

the riskier opportunity. So, the parameters $(\pi_{b,tech}^{serve}(\xi), \pi_{b,field}^{serve}(\xi), \pi_{b,tech}^{field}(\xi), \pi_{b,field}^{field}(\xi))$ are all set to 1 for all players ξ of a team. Analogously, the *safe* action is mapped to the parameters $(0, 0, 0, 0)$ for all players ξ of a team.

To describe the played s-MDP actions *final* and *prefinal* with adequate parameters of the basic decision rule, an estimation from the raw data has been made: For this purpose, all services and field attacks played in the finale match respectively in the pre-final matches have been classified. The classification has been done by the script `evaluate-gdata.js` that gets the raw data file `*.gdata` from the Beach Volleyball Tracker as an input and outputs a file named `*.gprobs.csv`. The output file `*.gprobs.csv` contains the estimated parametrization of the basic decision rule. The script and all data files can be found in the supplementary material provided in Appendix E. The estimates are the relative frequencies of a technique or a target field used in the g-data. For instance, Brink has made in the final match 26 services from which 1 landed in a border field and 25 in a non-border field. So, the parameter of Brink that describes the proportion of serves into a border field is

$$\pi_{b,field}^{serve}(P_1) = \frac{1}{26} \approx 0.04.$$

The counted absolute frequencies are outputted by the script `evaluate-gdata.js` when the function `calculateStrat(logToConsole)` is called with the argument `true`.

The parameter estimates for the *prefinal* strategy are based on the g-data files containing all hits in pre-final matches. As this includes several matches against different opponents, it is not reasonable to calculate how often a service was made on a certain player of the opponent team. Therefore, the parameter π_s is set to 0.5 in the g-MDP strategy *prefinal*.

Table 5.2 specifies the mapping $\tau_A(s^s, a^s) = \pi$. For simulating an s-MDP decision rule that, e.g., consists of a *risky* service and a *safe* field attack, Algorithm 7 can be applied. It combines the g-MDP decision rules given by the s-g-implementation τ_S and τ_A .

Having specified an s-g-implementation by the mappings τ_S and τ_A , it needs to be evaluated whether the g-MDP decision rules π reflect the s-MDP transitions. Therefore, in the next subsection, a validation of the s-g-implementation is made to check whether the s-g-implementation is meaningful.

5.2.2 Validating the Implementation

To validate whether the presented s-g-implementation of the last subsection is meaningful, s-MDP transition probabilities have to be compared with its estimates from a g-MDP simulation. Screenshots of the used g-MDP simulation can be found in Appendix B and the simulation itself in the supplementary material provided in Appendix E. The realization of g-MDP states is generated by Algorithm 7 that simulates a s-MDP decision rule in the g-MDP. The estimation of the s-MDP transition probabilities from a realization of g-MDP states is done according to Proposition 5.1.4.

However, the “real” s-MDP transition probabilities are also not known in sports games and must be estimated too. As described at the beginning of this chapter and visualized in Figure 5.2, appropriate data records for the s-MDP needs to fulfill a bunch of parameters. In particular, for this implementation of the 2-MDP approach for beach volleyball, data records must be from a match where the participating teams are Brink-Reckermann and Alison-Emanuel. This is the case since s-MDP transition probabilities depend, in contrast to the g-MDP transitions, on the abilities of both teams. So, the only match from which “real” s-MDP transition probabilities can be estimated is the final match of the Olympic beach

estimation method	P_{final}^{serve}	\bar{P}_{final}^{serve}	Q_{final}^{serve}	\bar{Q}_{final}^{serve}	P_{final}^{field}	\bar{P}_{final}^{field}	Q_{final}^{field}	\bar{Q}_{final}^{field}
realized probabilities final	2%	4%	4%	14%	49%	17%	55%	16%
simulating the g-MDP with								
skills of all matches except final	2%	14%	3%	15%	33%	16%	29%	21%
skills of all matches	2%	12%	4%	14%	37%	17%	40%	21%
skills of final only	1%	2%	6%	10%	47%	14%	54%	17%

Table 5.3: Validation of simulated s-MDP transition probabilities

volleyball tournament. The played s-MDP decision rule in the final match uses per definition only the *final* action. So, the only possibility to validate the s-g-implementation is to compare estimates of s-MDP transition probabilities under the *final* strategy.

The reader may remember that in Subsection 4.2.5 in Table 4.3 also estimates for s-MDP transition probabilities from the s-data for the *risky* and *safe* strategy are presented. However, the observations used for that estimations are a subset of the observations used for estimating the *final* strategy. So validating the s-g-implementation with the *risky* or *safe* strategy would contain the same but less data points while the same g-MDP mechanism is used.

Table 5.3 contains all estimates of s-MDP transition probabilities for the *final* strategy. The first line contains an estimate based on the realized transitions in the final match. These values are estimated from the collected s-data and the rounded version of the probability estimates presented in Subsection 4.2.5.

The last three lines of Table 5.3 contain estimates based on realizations of g-MDP states according to Proposition 5.1.4. The g-MDP was simulated using the decision rule π_{final} for both teams with different skill estimates. One estimation is based on 500 simulation runs with each containing 100 rallies. The different skill estimates have an impact on the probability estimates as the probability presented in one column varies. The last line of Table 5.3 corresponds to a g-MDP simulation in which the skills of all players are only estimated from the final match. So, these skills describe the performance of the players in the final match. Together with the *final* strategy, the last line of Table 5.3 is a line with “perfect knowledge” for the g-MDP. On average, the estimated probabilities of the last line are closest to those in the first line. The remaining difference can either be justified by the estimation fault of the “real” s-MDP decision probabilities presented in the first line or by not entirely perfect modeling the game mechanism in the g-MDP. Most likely it is a mixture of both.

The average absolute difference of the first and the fourth line is 2 percent points. In comparison when the skills are estimated from the pre-final matches, this average difference is 7.5 percent points.

The reader may ask whether this accuracy is good enough to assume a meaningful s-g-implementation? The validation results presented in Table 5.3 are of course gone through several iterations of validation and modifications of the g-MDP system dynamics. However, the g-MDP system dynamics were only adapted if there was a reason for it. For instance, as the data was collected from video sequences and not in an organized training session, the targeted field of a hit could only be guessed. And as described in Subsection 4.3.3, a deviation was only counted if the ball landed in an outside field. So, there existed a reason to adopt the system dynamics of the g-MDP such that a deviation will always lead to an outside field if the targeted field is a border field.

It should be noted that the g-MDP does not contain a general or global fitting parameter. The use of such a global fitting parameter may be discussed for some reasons. However, in this case, it was omitted since this could lead to an over-fitting to the final match which is also only one sample case for validation of the 2-MDP approach.

5.2.3 Answering Strategic Questions

Finally, the 2-MDP approach is applied to the strategic question raised for the s-MDP in Subsection 4.2.5. It should be evaluated whether high risk or safe play of Germany leads to a higher winning probability. The ultimate goal is to give a justifiable strategy recommendation prior to the final match.

As visualized in Figure 5.2, in the 2-MDP approach, the g-MDP is simulated to generate estimates for the s-MDP transition probabilities. These s-MDP transition probabilities of different s-MDP strategies are then evaluated by using the analytical formula for the s-MDP. The strategy with the highest winning probability is returned as the recommended strategy.

Skill estimation from pre-final matches						
strategy a	p_a^{serve}	\bar{p}_a^{serve}	p_a^{field}	\bar{p}_a^{field}	$\alpha_{a,b}^P$	winning Prob
<i>risky-risky</i>	5%	20%	42%	18%	45%	78%
<i>risky-safe</i>	5%	20%	18%	15%	39%	29%
<i>safe-risky</i>	1%	13%	42%	18%	47%	82%
<i>safe-safe</i>	1%	13%	18%	15%	40%	32%
<i>prefinal</i>	2%	16%	33%	16%	44%	68%
	q^{serve}	\bar{q}^{serve}	q^{field}	\bar{q}^{field}		
<i>prefinal</i>	2%	12%	27%	19%		

(a) pre-final setting

Skill estimation from final match						
strategy a	p_a^{serve}	\bar{p}_a^{serve}	p_a^{field}	\bar{p}_a^{field}	$\alpha_{a,b}^P$	winning Prob
<i>risky-risky</i>	3%	8%	54%	15%	34%	46%
<i>risky-safe</i>	3%	8%	40%	13%	32%	20%
<i>safe-risky</i>	1%	2%	54%	15%	35%	49%
<i>safe-safe</i>	1%	2%	40%	13%	32%	22%
<i>final</i>	1%	2%	47%	13%	34%	36%
	q^{serve}	\bar{q}^{serve}	q^{field}	\bar{q}^{field}		
<i>final</i>	6%	10%	54%	15%		

(b) post-final setting

Table 5.4: Estimation of s-MDP probabilities from g-MDP simulation

Table 5.4 contains the estimation results for the s-MDP transition probabilities together with their winning probabilities. Each estimation was again based on 500 runs with 100 rallies. The 95%-confidence interval of the g-MDP simulation with 500 batches with 100 runs is smaller than 0.01. As

the g-MDP is itself based on input probabilities, which are the skills of the players, two different types of simulations were carried out. In the upper half of Table 5.4, the skill estimates are based on pre-final matches while in the lower half the skills estimates are only based on the final match. So, if the strategic question shall be answered priori to the final match, the values in the lower half would not be available. Observe, that according to Theorem 4.2.3 the parameter $\alpha_{a,b}^P$ is sufficient to determine an optimal strategy for the s-MDP. The winning probability presented in the rightmost column is calculated by dynamic programming and gives supplementary information. The recommended strategy prior to the final match is to play *safe* in the serving situation and *risky* in the field attack situation. This strategy has a winning probability of 82% percent.

This strategy is also the recommended strategy a posteriori the final match. However, the winning probabilities decreased significantly to 49%. The reason for this change may lie in the different skill estimates used for both teams. The reader may compare the tables in Appendix A that contains for each player tables with skill estimates based on the final match and skill estimates based on all pre-final matches. The number of observations for estimating the skills solely from the final match is very small which can be seen by the values in the #-column of the corresponding table. In cooperation with a team, there are more reliable methods than analyzing video sequences of matches to estimate the player skills. For instance, specified training sessions could be designed. In this study, the skill estimations based on the final were made such that the a priori recommendation could be compared to an a posteriori recommendation while taking a higher estimation error into account.

Besides these variations in the performance of both teams, also the played strategy of the Brazilians changed from the pre-final matches to the final match. The differences between the *prefinal* and the *final* strategy of Brazil can be seen in Table 5.2. The proportion of jump serves made by Alison increased from 37% to 68% and the proportion of field attacks in a border field made by Emanuel increased from 31% to 54%. These are the two most significant differences that may result from the more challenging opponent Germany.

For completeness, Table 5.4 also contains the estimates for the *final* and *prefinal* strategy of Germany. The estimated values for the *final* strategy, based on skills of the final match, are generated in the same setting as those presented in the last line of Table 5.3. The minor differences occur because the values are generated in a different simulation run. The results show that with the recommended *safe-risky* strategy, the winning probabilities of Germany would have been higher.

5.2.4 Comparison of Results

After having answered the strategic question regarding the s-MDP decision rule two times, a comparison between the approaches will be made in this section. Handling the strategic question only by considering the set-SSO-MDP, which was done in Subsection 4.2.5, will in the following be called the *direct approach*. The results of the direct approach will be compared to those of the 2-MDP approach presented in the last subsection.

The two approaches are first compared qualitatively, which will be followed by a quantitative evaluation of the recommendations.

In the setting of this thesis, it is not possible to answer the strategic question prior to the final match with the direct approach. The reason is that no appropriate data records for the s-MDP exist before the final match. In contrast, the 2-MDP approach can be evaluated with skill estimates based on pre-final matches and the strategy *safe-risky* is recommended.

Obviously, a second SSO-MDP, the g-MDP, is needed for the 2-MDP approach. The g-MDP requires knowledge of the game mechanism and has to be validated. A posteriori the final match the 2-MDP approach recommends the same strategy (*safe-risky*). In contrast, the direct approach evaluates *safe-safe* has the best strategy.

For a quantitative comparison, an adequate measurement is needed. Ideally, the final match could be repeated several times with the two strategy recommendations. The strategy that yields the higher expected winning probability can then be evaluated as the better recommendation. However, it is not possible to replay the final match several times. So, another measurement is needed. The final match is the only match between Brink-Reckermann and Alison-Emanuel, and the only strategies that were realized in the final match are the *final* strategies. The *final* strategy can be described as a mixture of the *risky* and *safe* serves respectively field attacks. To measure the correctness of the estimated probabilities for the *risky* and the *safe* strategy, the transition probabilities of the *final* strategy will be calculated as a mixture of the estimated probabilities for *risky* and *safe*. The quantitative measure for both approaches is then the difference of the result of the mixed strategy probabilities and the realized transition probabilities of the final match.

Table 5.2 already contains a representation of the *final* strategy of the g-MDP as a mixture of the *risky* and the *safe* strategy. For instance, the final strategy played by Brink is a mixture of 39% jump services and 61% float services, which are the hitting techniques of the risky respectively safe service strategy.

For the s-MDP, the mixture of risky and safe service respectively field estimates played by the Germans can be found in Table 4.4. The proportion is specified as the relative amount of *risky* serves [field attacks] on all serves [field attacks] that can be classified as *risky* or *safe*. So, the *final-mix* played by Brink-Reckermann is:

$$final-mix_{serve} = \frac{1}{39} \approx 3\% \qquad final-mix_{field} = \frac{12}{23} \approx 52\%.$$

The estimated s-MDP transition probabilities for the *risky* respectively *safe* strategy are mixed with the proportion of the *final-mix*. The resulting s-MDP transition probabilities can be found in Table 5.5b.

Ignoring the fact that the pre-final matches are played against different opponents. All services and field attacks of the pre-final matches are also classified and estimations for *risky* and *safe* transition probabilities based on the pre-final matches are made. The values are presented in Table 5.5a. With these estimated s-MDP transition probabilities, the direct approach would have recommended playing *safe-risky*. This recommendation equals the recommendation of the 2-MDP approach. However, it is stressed out again that this result is a recommendation based on matches against different opponents. And therefore, this recommendation does not take into account which team is the opponent team in the final match.

Table 5.6b finally presents the realized probabilities of the final match together with the estimation based on a mixture of the *risky* and the *safe* strategy estimates. This procedure feels a little bit awkward but is the only way found to evaluate the estimates for the *risky* and *safe* strategy. The simulated values from the g-MDP are the same as already presented in the validation subsection of the g-MDP, see Table 5.3. The last column presents the average total difference between the realized probabilities and the estimated probabilities in the corresponding row. The variations of both approaches lie in the same magnitude. Only in the pre-final setting, the estimates resulting from the g-MDP simulation are

Based on pre-final matches								
strategy a	#	p_a^{serve}	\bar{p}_a^{serve}	#	p_a^{field}	\bar{p}_a^{field}	$\alpha_{a,b}^P$	winning Prob
<i>risky-risky</i>	32	16%	22%	58	66%	17%	37%	77%
<i>risky-safe</i>	32	16%	22%	33	48%	0%	36%	60%
<i>safe-risky</i>	102	4%	9%	58	66%	17%	34%	69%
<i>safe-safe</i>	102	4%	9%	33	48%	0%	32%	50%
<i>prefinal-mix</i>	134	7%	12%	91	59%	11%	34%	65%
	#	q^{serve}	\bar{q}^{serve}	#	q^{field}	\bar{q}^{field}		
<i>prefinal-mix</i>	165	2%	10%	105	58%	15%		

(a) pre-final setting

Based final match								
strategy a	#	p_a^{serve}	\bar{p}_a^{serve}	#	p_a^{field}	\bar{p}_a^{field}	$\alpha_{a,b}^P$	winning Prob
<i>risky-risky</i>	1	0%	0%	12	42%	25%	29%	14%
<i>risky-safe</i>	1	0%	0%	11	64%	9%	34%	73%
<i>safe-risky</i>	38	3%	0%	12	42%	25%	30%	18%
<i>safe-safe</i>	38	3%	0%	11	64%	9%	36%	77%
<i>final-mix</i>	39	3%	0%	23	52%	17%	33%	44%
	#	q^{serve}	\bar{q}^{serve}	#	q^{field}	\bar{q}^{field}		
<i>final-mix</i>	28	4%	14%	39	59%	15%		

(b) post-final setting

Table 5.5: Direct estimation of s-MDP probabilities

significantly worse. However, in the post-final setting, the estimated transition probabilities from the g-MDP outperform the direct approach.

Using the 2-MDP approach opens up new possibilities that are not available from a direct approach. Two of these new possibilities are presented in the following subsections.

5.2.5 Sensitivity Analysis

The strategic recommendations made by the 2-MDP approach are based on input probabilities that are only estimates. Since any estimate incorporates some error, also the output probabilities of the 2-MDP approach is affected by these errors. A sensitivity analysis can help to recognize whether an error in the input data can lead to a fundamentally different recommendation.

A *Strategy-Skill-Score-Card* (SSSC) is a tool that is generated by the 2-MDP approach and visualizes two sensitivity analyses in one diagram. It may take some time for a practitioner or coach to understand an SSSC completely. However, if the structure is recognized, the information provided by a card can be gathered fast.

In principle, an SSSC determines the preferred strategy of two given strategies in a particular situation. In all example SSSCs presented in this subsection, the risky strategy (risky service and risky

estimation method	data	dec. rule	p^{serve}	\bar{p}^{serve}	q^{serve}	\bar{q}^{serve}	Avg. L^1 -Error
realized probabilities final			2%	4%	4%	14%	-
simulating the g-MDP	pre-final skills	<i>final</i>	2%	14%	3%	15%	3.19%
direct estimations	pre-final matches	<i>final-mix</i>	4%	9%	2%	12%	2.86%
simulating the g-MDP	final skills	<i>final</i>	1%	2%	6%	10%	2.34%
direct estimations	final matches	<i>final-mix</i>	3%	0%	4%	14%	1.10%

(a) s-MDP transition probabilities for serving situation.

estimation method	data	dec. rule	p^{field}	\bar{p}^{field}	q^{field}	\bar{q}^{field}	Avg. L^1 -Error
realized probabilities final			49%	17%	55%	16%	-
simulating the g-MDP	pre-final skills	<i>final</i>	33%	16%	29%	21%	11.74%
direct estimations	pre-final matches	<i>final-mix</i>	57%	9%	60%	17%	5.71%
simulating the g-MDP	final skills	<i>final</i>	47%	14%	53%	17%	1.63%
direct estimations	final matches	<i>final-mix</i>	52%	17%	59%	15%	2.27%

(b) s-MDP transition probabilities for field attack situation.

Table 5.6: Comparison between 2-MDP approach and direct approach

field attack) is compared to the safe strategy (safe service and safe field attack) by subtracting the winning probability of the risky strategy from the winning probability of the safe strategy. If the result is a positive value, which is encoded by a green color, the *safe* is the recommended strategy. A negative value is encoded by a red color and means that *risky* is the preferred strategy. Yellow color means that both strategies have a very similar winning probability.

As mentioned before, an SSSC simultaneously visualizes two sensitivity analyses. In one analysis, the skills of the considered team are parametrized for a certain hitting technique. Thereby, the individual success probability $p_{succ,g}$ (*field, tech*) and the individual fault probability $p_{fault,g}$ (*field, tech*) are varied from 0 to 1 in 0.1-steps. Those two probabilities determine the probability of a deviation $p_{dev,g}$ (*field, tech*). In the four example SSSCs presented in this thesis, the smashing skills of both players of team P are varied in Figure 5.3, the skills for a planned shot in Figure 5.4, the skills for a jump serve in Figure 5.5, and finally, the skills for a float serve in Figure 5.6. These techniques are chosen since they are the main characteristics of the *risky* and the *safe* strategy. As described in Table 5.2, a jump serve and an attack hit are played in the *risky* strategy while a float serve and a planned shot are performed in the *safe* strategy.

The other sensitivity analysis concerns the s-MDP probabilities of the opponent team. Hereby, the direct point probability q^{field} and the fault probability \bar{q}^{field} of a field attack of team Q are varied from 0 to 1 in 0.2-steps. The two probabilities determine the probability \hat{q} that describes the probability of a subsequent field attack.

These two parameterizations are captured in one graphic in the following way: An SSSC is a large triangular chart that consists of several smaller triangular chart. In each smaller chart, the s-MDP transition probabilities of the opponent team are fixed to values determined by the location of the smaller chart. Inside a small chart, the skills of the considered team are varied. Each square inside a small

chart results from a simulation with 500 batches with 100 rallies and corresponds to a skill setting of the considered team and an opponent characterized by the s-MDP transition probabilities. The simulation is carried out with the pre-final skill estimations of both teams. The color of the square represents the preferred strategy in that setting.

An SSSC should be read by a coach or a practitioner in the following way: First, the opponent team needs to be classified concerning its s-MDP transition probabilities. For instance, “the probability of a direct point after a field attack is around 0.4 and the probability of a fault also at 0.4”. So, the coach should look at the smaller chart in the third row and the third column counted from the origin. Then, the coach needs to estimate the skills of his team. For instance, “my team can normally do a smash with a success rate of 0.4 and a fault rate of 0.5”. In this case, the coach has to consider the square in the fifth row and the sixth column – again counted from the origin –, and sees that the *safe* strategy is preferred to the *risky* strategy. The coach may ask “What if my team performs better today and can achieve a 0.5 success rate for the smash with the same fault rate?”. He may have a look at the same chart and finds that still *safe* is the preferred strategy. All squares in the neighborhood of the original square under consideration are green, so he knows that the *safe* strategy is quite robust to skill changes of his team.

Furthermore, it is also possible to examine a scenario where the opponent’s characteristic changed. For instance, in our example, a stronger opponent with a fault probability of 0.2 corresponds to the smaller chart in the third row and the second column. This smaller chart is more yellow than green. So, against a stronger opponent the *safe* strategy does not outperform the *risky* strategy anymore. The reader may have a look at Figure 5.3 to recover these findings.

After the general introduction to SSSCs, the presented figures are analyzed in more detail. In Figure 5.3, the smashing skills of Brink-Reckermann are parametrized. Since the figure is generally yellow or green, it can be concluded that most of the time it does not matter whether *risky* or *safe* is played and otherwise *safe* is the better strategy. Only if Germany can play the smash excellent (success rate larger than 0.9), there exist some opponents where a *risky* strategy is preferred. By the lines in the smaller charts, the real values of smashing skills of Germany are presented. It can be seen that their smashing skills lie around 0.9. Therefore, it may be valuable for them to play a smash which corresponds to the recommendation given in Subsection 5.2.3. In Table 5.4 for the pre-final setting the *risky-risky* strategy is preferred to the *safe-safe* strategy. Note that the SSSCs are generated for the pre-final setting and compare the strategies *risky-risky* to *safe-safe*.

In Figure 5.4, the planned shot, which is the counterpart of the smash, is parametrized. The figure is only yellow or red which means that there does not exist any configuration where a *safe* strategy is preferred to the *risky* one. The powerful smashing skills of Germany can explain the dominance of one strategy. A smaller chart at the top corresponds to a strong opponent whose direct point probability is very close to 1. In Figure 5.4, those smaller charts are mainly yellow, which means that the winning probability of both strategies is very similar. The opponent is so strong that the strategy choice has no impact on the winning probability – which is very low. A smaller chart at the right corresponds to a very weak opponent. In Figure 5.4, there also occur more yellow squares the more one looks at the bottom right. In those cases, the opponent is so weak that the strategy choice will also have no impact on the winning probability. For the opponent Brazil, the pre-final direct point probabilities are $q^{field} = 0.27$ and $\bar{q}^{field} = 0.19$, compare Table 5.4a. These direct point probabilities lead to the smaller chart in the second column and the second or third row – since the step size is 0.2 the real value lies between the two drawn charts.

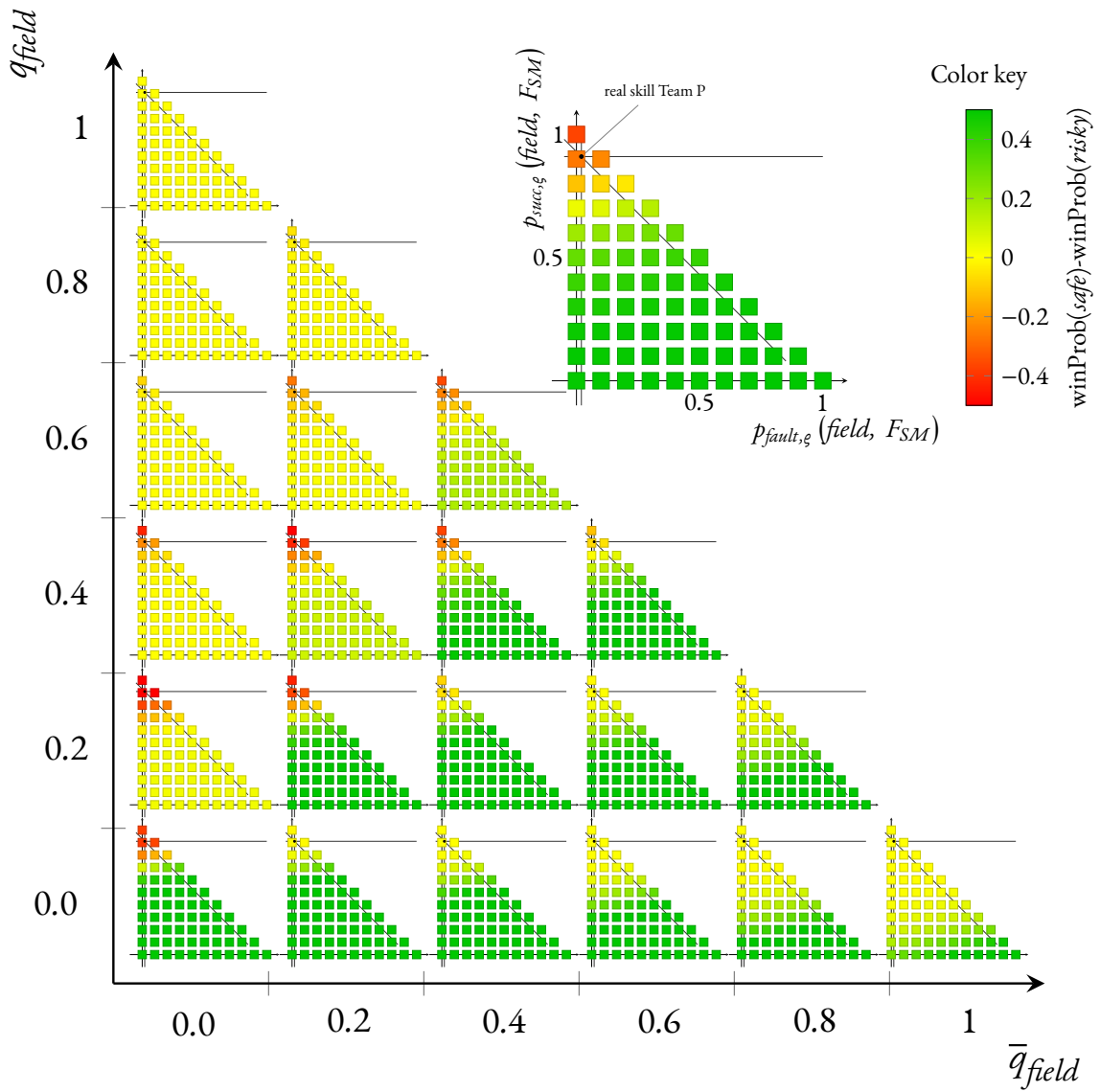


Figure 5.3: Skill-Strategy Score Card: *safe* versus *risky* play for varying smash skills (FS)

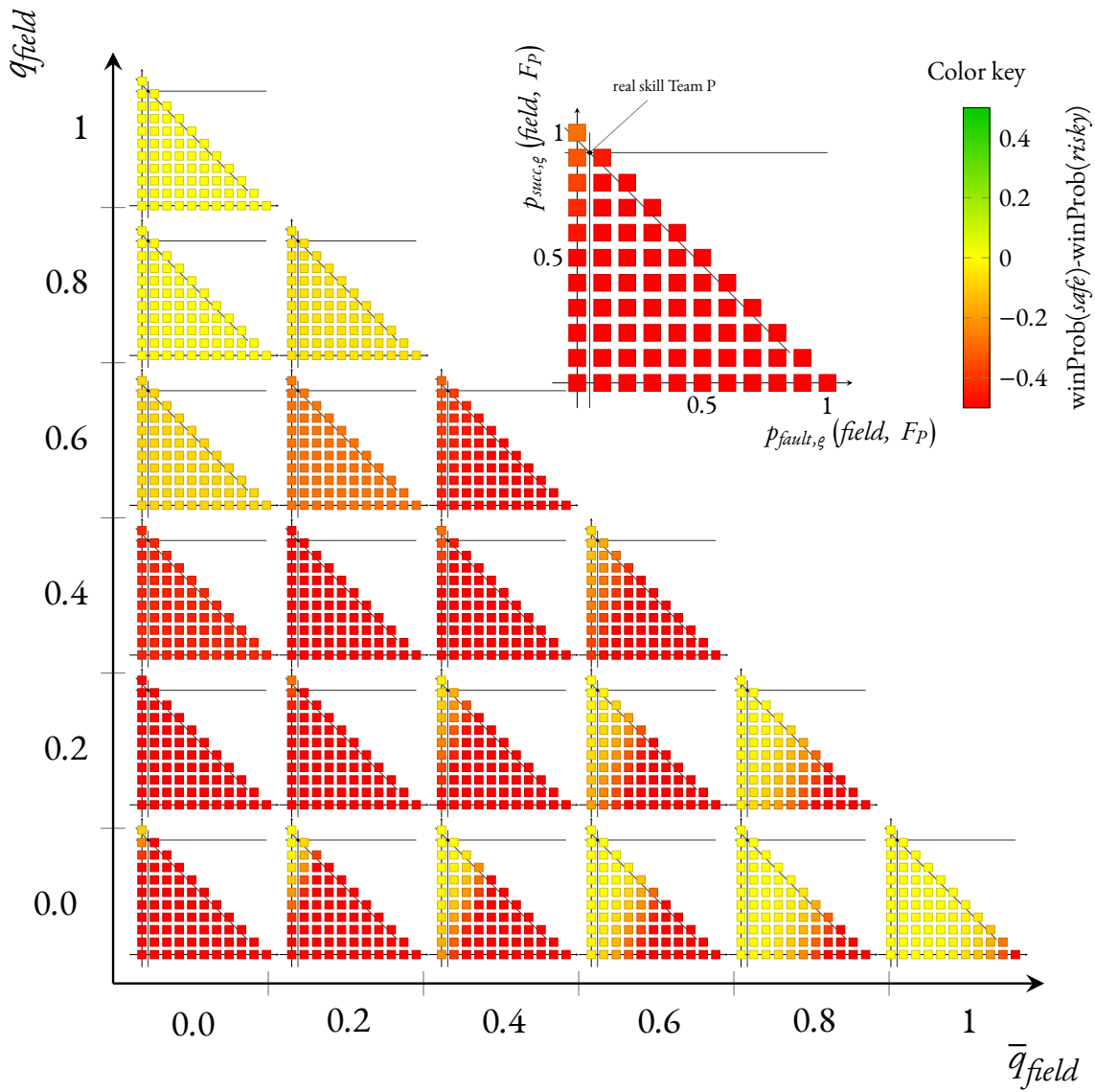


Figure 5.4: Skill-Strategy Score Card: safe versus risky play for varying shot skills (FP)

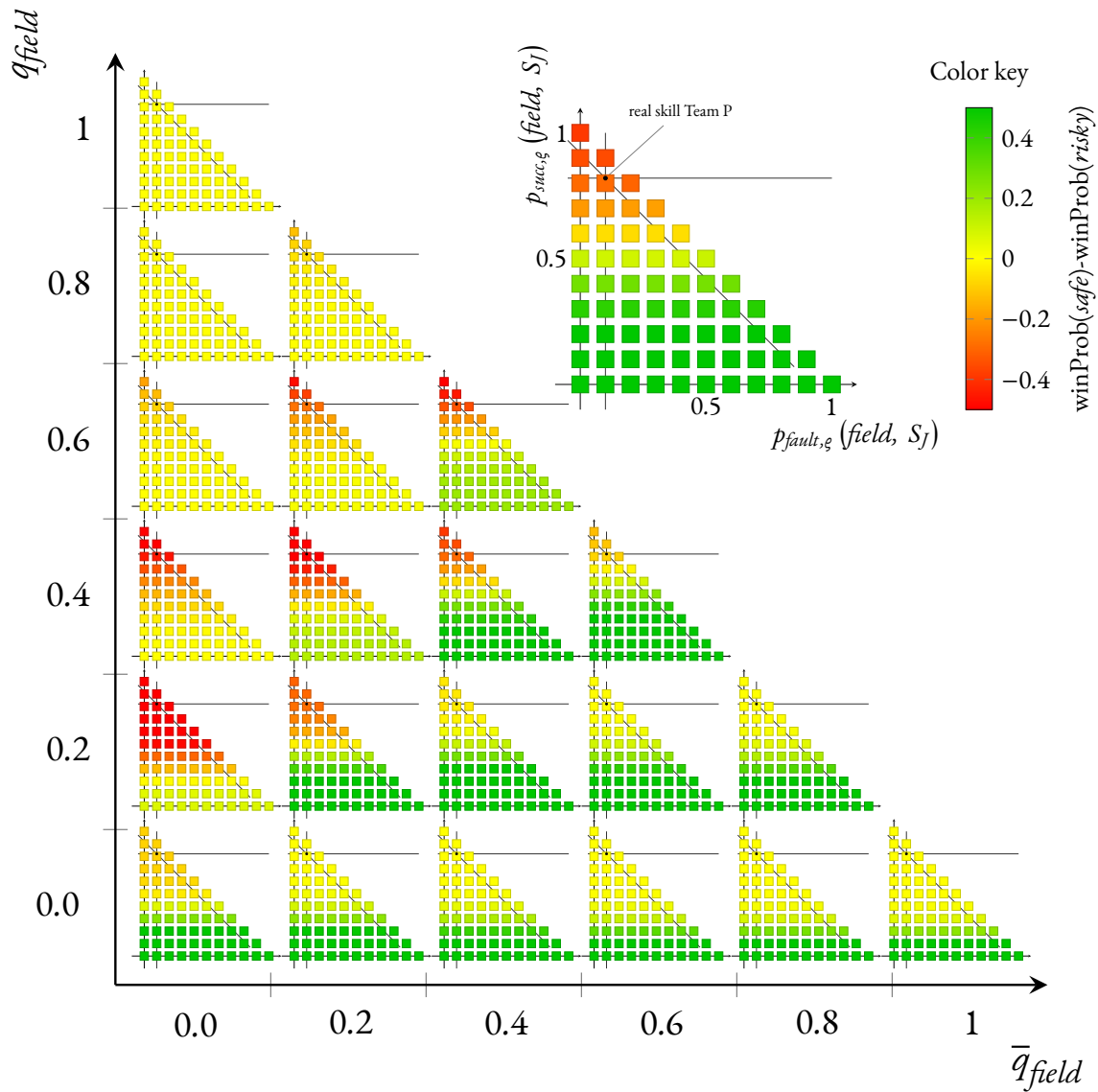


Figure 5.5: Skill-Strategy Score Card: *safe* versus *risky* play for varying jump serve skills (S_J)

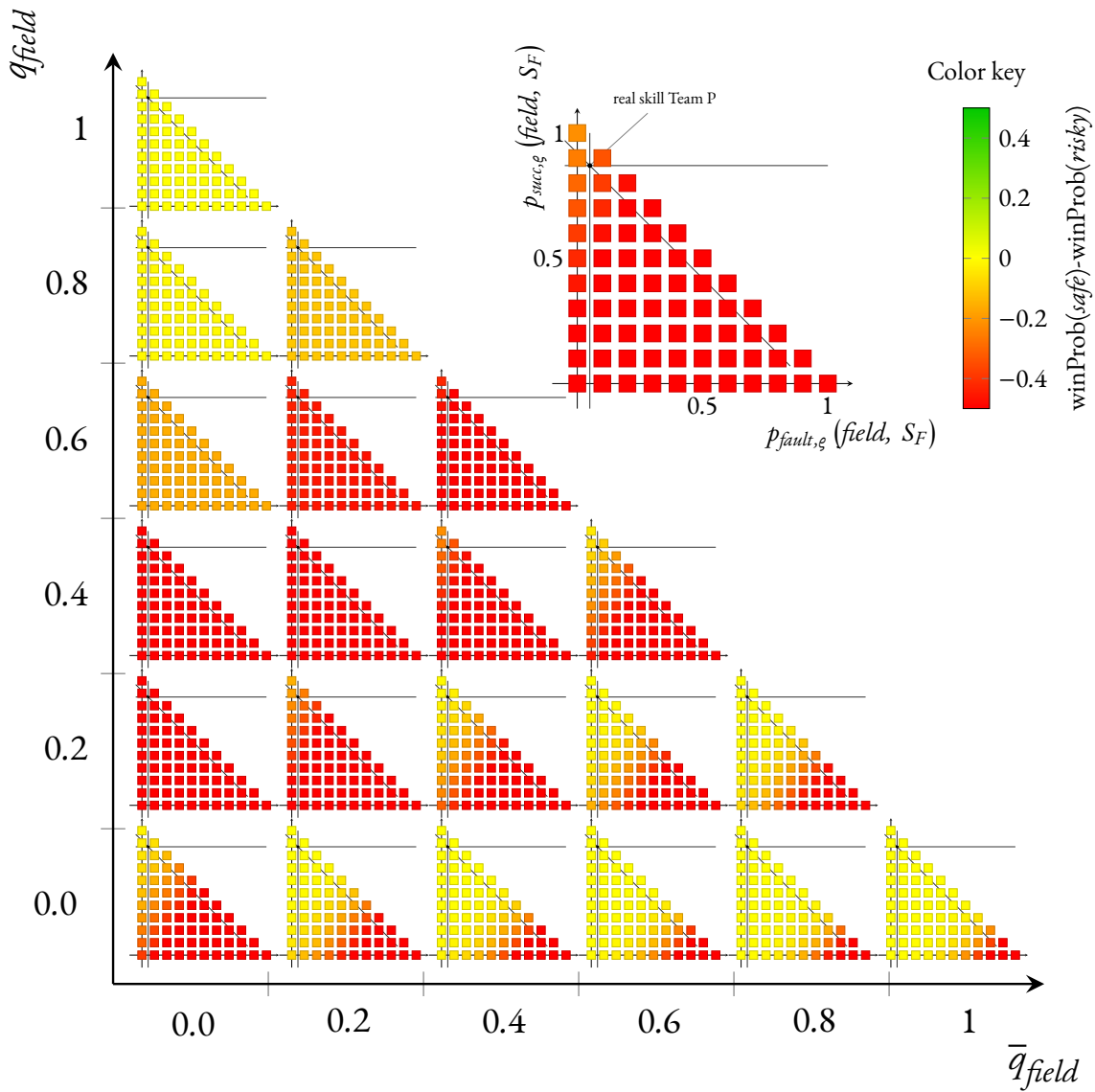


Figure 5.6: Skill-Strategy Score Card: *safe* versus *risky* play for varying float serve skills (S_F) and $p_{fault,\epsilon}(field, S_F)$

In Figure 5.5 and Figure 5.6, an analysis of the serving skills of Brink-Reckermann is done. Similar findings to those of the last two figures can also be made here. For the jump serve, the transition, where the preferred strategy changes, seems to be sharper and lies more central in the smaller charts. So, for a success probability of 0.7 for the jump serve, the *risky* service can be preferred against some opponents.

In the smaller charts of hitting techniques that correspond to the *safe* strategy (float serve and planned shot), a principle difference to the *risky* techniques can be seen: The gradient of the color changes is not parallel to one of the axis. Instead, the gradient's slope has an angle of circa 45 degrees. In contrast, in a smaller chart of a smash or a jump serve, this gradient is parallel to the x-axis. These findings can be interpreted in the following way: For a safe hit, a deviation is nearly as good as a success, while for a risky hit a deviation is as bad as a fault. However, this finding can be explained by the fact that in the *risky* strategy a risky hit is always played in a border field, while a safe hit is played in a non-border field.

5.2.6 Two-Person-Constant-Sum Game

This subsection introduces constant-sum-matrix-games that are generated by the 2-MDP approach. A constant-sum-matrix-game encodes the winning probability of a team for different g-MDP parameterizations of their own and the opponent's strategy. By changing the strategy of both teams, the scope of Markov decision processes is left towards game theory. The game-matrix can be used to identify Nash equilibriums. Furthermore, by detecting graphic patterns, it is possible to identify dominating parameter settings.

Table 5.7 and Table 5.8 present a constant-sum-matrix-game for the final match of the Olympic games 2012 between Brink-Reckermann and Alison-Emanuel. They are generated by the 2-MDP approach implementation defined in this section. The difference between both tables is that Table 5.7 contains results generated by a g-MDP in which all skill estimates are based on pre-final matches while in Table 5.8 the final match was used as a data basis for the skill-estimates.

Each colored, small square corresponds to a simulation of the g-MDP with 100 batches with 100 rallies each. The color of a square represents the winning probability of Germany. A green colored square means that Germany will win the match with a probability of 100%, a red colored square means that Germany loses with a probability of 100% and yellow indicates close to equal winning probabilities of both teams. Note, that in contrast to SSSCs the color corresponds to the winning probability in a particular strategy setting and not to the difference of the winning probabilities of two distinct strategies.

The played strategy combinations of both teams are encoded at the sidelines by white and gray squares. Each square corresponds to one parameter of the basic decision rule. A white square corresponds to the value 0 and a gray square to a value of 1. In the presented game-matrices, the following five parameters are alternating between 0 and 1:

$$\left(\pi_b, \pi_{b,*}^{serve}(\varrho_1), \pi_{b,*}^{field}(\varrho_1), \pi_{b,*}^{serve}(\varrho_2), \pi_{b,*}^{field}(\varrho_2) \right). \quad (5.1)$$

Observe that the used technique and the target field of one situation are combined in one parameter. The serving strategy π_s is in the pre-final setting is fixed to 0.5 and in the post-final setting to the observed value of π_s in π^{final} . By alternating these five parameters between 0 and 1, the resulting matrix is a 32×32 -matrix that contains the most extreme strategies. For example, the first line in both tables corresponds to Germany playing the strategy 0, 0, 0, 0, 0, which means that player 2, is always blocking and both players play a safe service, and a safe field attack.

The small square in the upper left corner of both tables visualizes the winning probability of Germany for the strategies played by both teams in the final match.

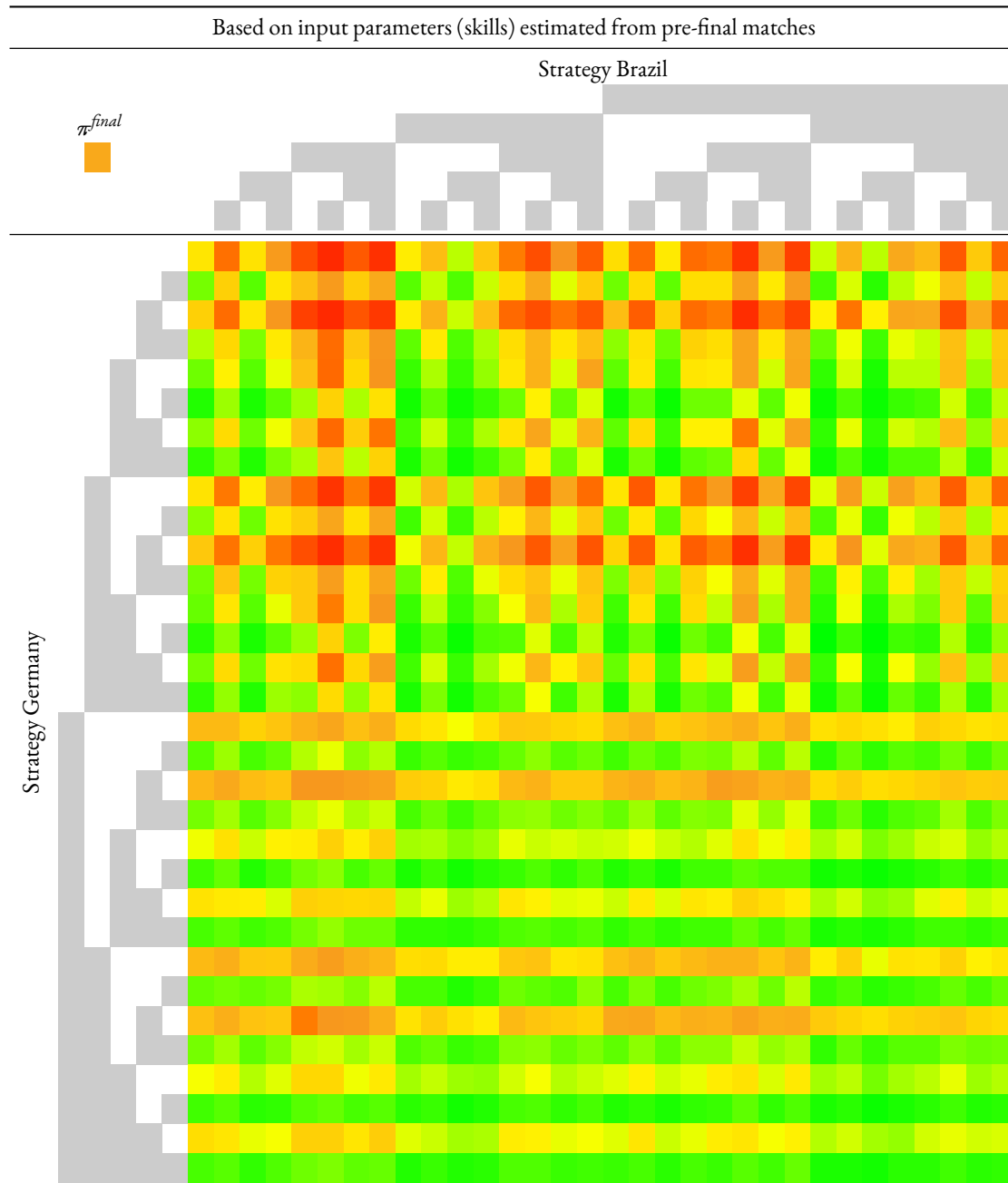


Table 5.7: Winning probabilities of Germany regarding different strategy combinations in the pre-final setting.

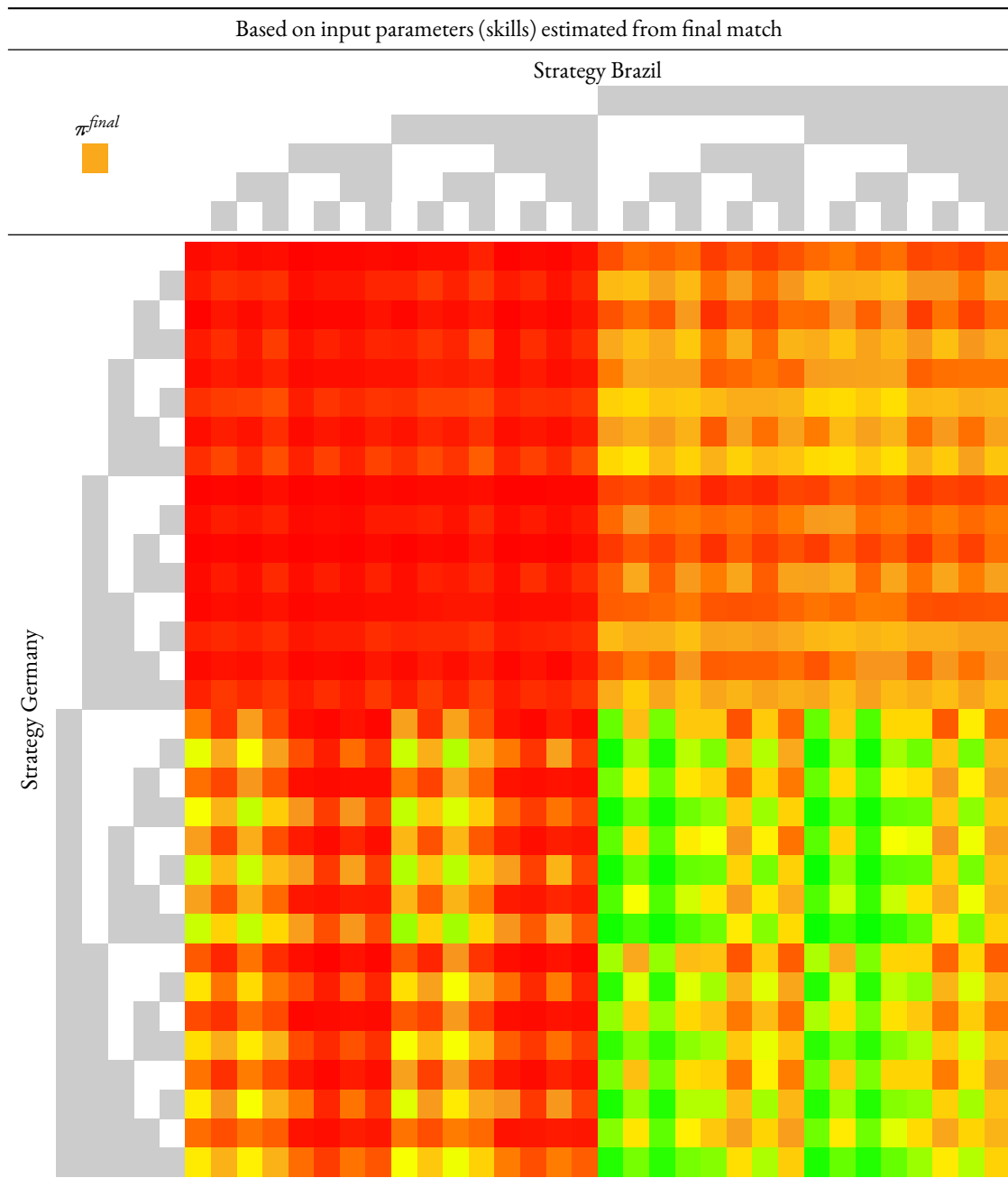


Table 5.8: Winning probabilities of Germany regarding different strategy combinations in the post-final setting.

Looking at the tables from a bird's eye perspective, one may detect patterns in both tables. These patterns lead back to parameter values that have a major influence on the winning probability. Depending on the frequency of the pattern, and the ordering of the alternated parameters, the corresponding

parameter can be determined.

For instance, in Table 5.7, every second row is a bit greener than the other rows. As this pattern occurs every second row, it can be concluded that the last parameter of the tuple in line 5.1 is responsible for this pattern. The last parameter is $\pi_{b,*}^{field}(P_2)$ such that the pattern can be interpreted in the following way: If Reckermann, who is player 2 of Germany, plays a risky field attack, i.e., a smash into a border-field, it has a positive impact on the winning probability of Germany.

Another example is the left half of Table 5.8, which seems to be redder than the right half. The first parameter π_b of Brazil's strategy is responsible for this pattern. The left half corresponds to a value of 0 for the parameter π_b . It can be concluded that it is promisingly for Brazil that Emanuel (Player 2) is the blocking player. The recommendation of Emanuel as the blocking player is an exciting finding since Alison is the taller player that goes more often to the block (96%). However, this finding may be an artifact of the estimation of Emanuel's blocking skills. Table A.14 shows that the estimation of Emanuel's blocking skills is based on two observations in the post-final setting.

As the lower half seems to be more green than the upper half, the game-matrix recommends setting $\pi_b = 1$ for the German team. So Brink is suggested to be the blocking player. Since Reckermann is, in general, the blocking player of Germany, this is also a similarly surprising result as for the Brazilians. This finding may also be an artifact of the small estimation basis for Brink's blocking skills in the post-final setting, compare Table A.2.

However, even if Brink has done fewer blocks than Reckermann and Emanuel fewer blocks than Alison, this does not mean that their estimated skills are ordered in the same manner. For example, Brink has a direct point rate after a block of 0.2 in the pre-final matches while Reckermann has only a direct point rate of 0.12. It is not possible to determine just from the blocking skill estimates the "better" blocking-player of both teams since there are opposite effects. An opposing effect may be that one player has a higher direct point probability, but he also has a higher fault or no-block probability. So, by only looking at the input-skills the block recommendation of the game-matrix cannot be validated nor refuted. Of course, it has to be carefully checked that these findings are no artifact of the g-MDP. But, in a general view, this finding shows the benefit of an MDP: Sometimes it is not clear to state solely from the data which of two opposing effects is dominating the other. This trade-off can be evaluated by an MDP to come to a result.

In general Table 5.8 is redder than Table 5.7, this indicates a lower winning probability of Germany when the skills are estimated from the final match. A reason for the lower winning probability may be that Germany's performance dropped while Brazil's performance improved in the final match. Another reason may be that the teams did not need to show their full potential in the pre-final matches such that the Brazilian's skills are underestimated from the pre-final matches.

The most promising strategy for Germany in Table 5.8 is the strategy

$$\begin{aligned}\pi_b &= 1, \\ \pi_{b,*}^{serve}(P_1) &= 0, \\ \pi_{b,*}^{field}(P_1) &= 1, \\ \pi_{b,*}^{serve}(P_2) &= 1, \\ \pi_{b,*}^{field}(P_2) &= 1.\end{aligned}$$

This strategy recommendations matches to the recommendation of *safe-risky* for the final match from Subsection 5.2.3.

The presented example findings showed, how general patterns and essential parameters could be identified with the help of a constant-sum-matrix-game. Without a 2-MDP approach, it would likely not be possible to generate a table with 32×32 -strategy combinations from solely one tournament or one season. Furthermore, it could be that some of these strategy-combinations have never been played in reality, but a coach wants to investigate some hypothetical scenarios, which is possible with a 2-MDP approach.

Chapter 6

Conclusion and Outlook

After five chapters with content related to sports strategy optimization, the most important results are repeated together with an assessment in the next paragraphs:

The theoretical results regarding MDPs that model a sport-strategic question are presented in Chapter 3. Definition 3.2.2 introduces a class of MDPs, called SSO-MDPs, that is suitable for sports strategy optimization. Assumption 3.2.1 is identified as the underlying characteristic of SSO-MDPs. In Figure 3.2, SSO-MDPs are set in relation to other known SSO-MDPs.

The definition of an MDP class for sports strategy optimization problems provides a basis for general findings regarding MDPs modeling a sports game. The existing literature focuses in most of the cases on a special MDPs – one exception is Walker, Wooders, and Amir (2011) who analyze general Markov Games that satisfy a specific monotonicity criterion. However, the definition of a class of Markov decision processes is novel and an essential step towards findings that apply to a complete class of MDPs.

The most important findings for SSO-MDPs, presented in this thesis, are: The optimality equations have a unique fixed point (Theorem 3.4.2). All policies of SSO-MDPs are proper (Theorem 3.4.5). The dynamic programming operator is a contraction mapping (Theorem 3.4.6). Furthermore, a primal and a dual linear programming formulation of SSO-MDPs has been specified (primal LP for SSO-MDPs respectively dual LP for SSO-MDPs).

As the dual linear programming formulation can be interpreted as a static maximum flow problem, Section 3.6 investigates the relationship between a dynamic SSO-MDP and its static flow network formulation. It has been shown that feasible solutions of the time expanded maximum flow formulation (Problem 3.9) characterize decision rules (Theorem 3.6.16) of the corresponding SSO-MDP. Furthermore, the value of a feasible solution of the time expanded linear program corresponds to the value of the derived stationary policy (Corollary 3.6.17). The static maximum flow problem is defined in 3.6.18. A feasible solution of the static maximum flow problem can be derived from a feasible solution of the time expanded maximum flow problem (Proposition 3.6.19) and the other way around (Proposition 3.6.20). The objective values are identical up to a constant that is defined by the distribution of the starting state. The main result of Section 3.6 is Theorem 3.6.23 which combines all previous results to prove that the static maximum flow problem can be used to characterize an optimal policy of an SSO-MDP.

This result is an independent proof of the validity of the dual linear programming formulation for SSO-MDPs. It does not rely on the primal linear programming formulation for positive bounded

MDPs and duality theory which have been used in Subsection 2.3.7 to derive the linear programming formulations. The result of Theorem 3.6.23 is not new. However, the different approach gives some insights. For instance, in the proof of Proposition 3.6.20, the properties of SSO-MDPs are explicitly utilized. It explains why a static maximum flow formulation cannot be used for general MDPs to determine an optimal policy.

Since value iteration terminates faster for SSO-MDPs with an underlying acyclic graph, Section 3.7 provides an algorithm (Algorithm 6) that transforms a graph associated with an SSO-MDP (Definition 3.6.1) into an acyclic graph. In Bertsekas (2001), it is sketched how self-transitions can be eliminated. The generality and the degree of detail of Algorithm 6 is a significant improvement to be able to implement a transformation algorithm for arbitrary SSO-MDPs.

Chapter 4 and Chapter 5 involve findings from the application of SSO-MDPs to beach volleyball. First, two new SSO-MDPs for beach volleyball are presented in Chapter 4. The definition of the set-SSO-MDP is summarized in Table 4.1. It is a coarse model whose optimal policies can be determined analytically (Theorem 4.2.3). This analytical result is compatible with that of Walker, Wooders, and Amir (2011) but was developed independently. Also, the winning probability of the tie game can be computed from the input probabilities, see Subsection 4.2.4. In Subsection 4.2.5, the set-SSO-MDP is applied to the final match of the beach volleyball tournament of the Olympic games 2012 in London. After an evaluation of the analytical findings, the set-SSO-MDP is used to answer a strategic question. The results are presented in Table 4.4.

The second SSO-MDP, the rally-SSO-MDP, is a very complex model with a large state space. It is defined in Subsection 4.3.1. Due to the huge state and action sets, it is not possible to define those sets explicitly. Functions that specify conditions on the combination of actions or evaluate independent parts of the transitions are introduced to be able to describe the action sets and the system dynamics. Furthermore, the concept of a parameterized basic decision rule is introduced in Subsection 4.3.2. A parameterized basic decision rule eliminates unreasonable strategies and helps to focus on particular strategic decisions. The rally-SSO-MDP is one of the largest MDPs contained in the literature related on sports strategy optimization. To handle such a huge MDP new concepts were developed that may also be useful for other large SSO-MDPs.

To apply the rally-SSO-MDP to a real match, a massive amount of data needed to be collected and processed to get valid and useful input data. This part of work is described in Subsection 4.3.3. It involved a lot of time and required the development of software tools. Furthermore, the use of real data without direct contact to the teams led to new challenges. For instance, an aggregation scheme has been defined to be able to get estimates for all required input probabilities. Ideally, a sport strategic decision is addressed together with a team. A cooperation with a team leads to new possibilities for the generation of adequate input data. In this thesis, however, all data had to be collected from publicly accessible material.

The evaluation of a rally-SSO-MDP is, in this thesis, limited to a simulation of the model. Due to the size of the rally-SSO-MDP, no optimal policy could be determined with standard methods for infinite-horizon Markov decision problems.

After analyzing two SSO-MDPs of different granularity, a new method, called 2-MDP approach, is proposed in Chapter 5. The 2-MDP approach tries to combine the advantages of both granularities. A general description of the method is presented in the first subsection of the chapter, while the next section, Section 5.2, implements the 2-MDP approach for the two beach volleyball SSO-MDPs. One of

the biggest challenges of this thesis was to validate the 2-MDP approach. Since each match is just a very small sample and it is not possible to repeat a match several times, all evaluations and comparisons must be considered with caution. The main results of the 2-MDP approach can be found in Table 5.4, where the strategic question of Section 4.2.5 is answered a second time by the new approach. A comparison between the results of the approaches is presented in Table 5.6b. The skill estimates for the pre-final and the final setting, used in the g-MDP simulation, were not as static as assumed. However, in the final context, only one match was used as an estimation basis. One match contains probably far too few observations for reliable skill estimates. Hopefully, this weakness can be eliminated if there exists direct contact with the teams and therefore a deeper understanding of their skills.

Finally, Subsection 5.2.5 and Subsection 5.2.6 present two new tools that can be helpful for coaches and players: A skill strategy score card presents two sensitivity analyses in one chart, see Figures 5.3, 5.4, 5.5 and 5.6. The constant-sum matrix games in Tables 5.7 and 5.8 help to identify crucial parameters of a strategy. These new tools can be generated by the 2-MDP approach. Feedback from coaches and practitioners regarding those tools would be exciting.

Further research that could follow this work could be an application of SSO-MDPs to other sports games. Especially, applying the 2-MDP approach to a less structured type of sports, like handball or soccer, would be interesting. Regarding the SSO-MDPs of this thesis, it may be interesting to apply methods like column generation to the rally-SSO-MDP and test whether it is possible to determine an optimal policy.

Hopefully, research in other fields like automatic video tracking will help to grow the field of sports strategy optimization further. If there exists easier access to suitable databases, there will emerge more mathematical optimization problems. The tools and solutions provided by mathematical optimization will help the coaches and teams to extract valuable information from those data records.

Appendix A

Rally-SSO-MDP: Skill Estimates

This chapter presents supplementary g-data to Subsection 4.3.3. For each player, the observations are split up into events observed in pre-final matches and events from the final match only.

A.1 Brink

target fields performance		#	Q11-Q14		#	Q21-Q24		#	Q31-Q34	
			<i>succ</i>	<i>fault</i>		<i>succ</i>	<i>fault</i>		<i>succ</i>	<i>fault</i>
Serve										
S_F	Po1 - Po4	11	1.00 (1.00)	0.00 (0.00)	10	1.00 (1.00)	0.00 (0.00)	-	-	-
S_J		4	0.81 (0.75)	0.00 (0.00)	1	0.89 (1.00)	0.00 (0.00)	-	-	-
Attack-Hit										
FS	out	0	0.94 (-)	0.00 (-)	0	0.94 (-)	0.00 (-)	-	-	-
	P11-P14	0	0.94 (-)	0.00 (-)	0	0.94 (-)	0.00 (-)	-	-	-
	P21-P24	10	0.99 (1.00)	0.00 (0.00)	4	0.87 (0.75)	0.00 (0.00)	-	-	-
	P31-P34	2	0.95 (1.00)	0.00 (0.00)	1	0.95 (1.00)	0.00 (0.00)	-	-	-
FE	out	0	0.75 (-)	0.11 (-)	1	0.76 (1.00)	0.11 (0.00)	-	-	-
	P11-P14	0	0.75 (-)	0.11 (-)	0	0.75 (-)	0.11 (-)	-	-	-
	P21-P24	1	0.67 (0.00)	0.11 (0.00)	2	0.68 (0.50)	0.20 (0.50)	-	-	-
	P31-P34	0	0.75 (-)	0.11 (-)	0	0.75 (-)	0.11 (-)	-	-	-
FP	out	0	0.95 (-)	0.02 (-)	0	0.95 (-)	0.02 (-)	0	0.95 (-)	0.02 (-)
	P11-P14	0	0.95 (-)	0.02 (-)	0	0.95 (-)	0.02 (-)	0	0.95 (-)	0.02 (-)
	P21-P24	1	0.96 (1.00)	0.01 (0.00)	5	1.00 (1.00)	0.00 (0.00)	0	0.95 (-)	0.02 (-)
	P31-P34	0	0.95 (-)	0.02 (-)	0	0.95 (-)	0.02 (-)	0	0.95 (-)	0.02 (-)

Table A.1: Input data from final match: Julius Brink – Serves and Attack-Hits

attack strength		<i>normal</i>			<i>hard</i>		
	performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>
Defense	<i>d</i>	4	0.57 (0.75)	0.38 (0.25)	9	0.36 (0.33)	0.54 (0.56)
	<i>d_m</i>	16	0.25 (0.25)	0.75 (0.75)	16	0.25 (0.25)	0.75 (0.75)
Reception	<i>r</i>	1	0.59 (0.00)	0.23 (1.00)	1	0.59 (0.00)	0.14 (0.00)
	<i>r_m</i>	11	0.91 (0.91)	0.00 (0.00)	3	0.95 (1.00)	0.00 (0.00)
Set	<i>s</i>	40	0.98 (0.98)	0.00 (0.00)	-	-	-
	performance	#	<i>block-point</i>	<i>block-ok</i>	<i>block-fault</i>	<i>noblock</i>	
Block	<i>b</i>	1	0.00	0.00	0.00	1.00	

Table A.2: Input data from final match: Julius Brink – Defense, Reception, Set, Block

target fields		Q11-Q14			Q21-Q24			Q31-Q34		
performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	
Serve										
<i>S_F</i>	Po1 - Po4	34	0.88 (0.88)	0.00 (0.00)	43	0.88 (0.88)	0.12 (0.12)	-	-	-
<i>S_J</i>		34	0.94 (0.94)	0.00 (0.00)	16	0.75 (0.75)	0.19 (0.19)	-	-	-
Attack-Hit										
<i>FS</i>	out	0	0.86 (-)	0.02 (-)	0	0.86 (-)	0.02 (-)	-	-	-
	P11-P14	0	0.86 (-)	0.02 (-)	0	0.86 (-)	0.02 (-)	-	-	-
	P21-P24	55	0.85 (0.85)	0.04 (0.04)	17	0.94 (0.94)	0.00 (0.00)	-	-	-
	P31-P34	7	0.77 (0.71)	0.01 (0.00)	2	0.89 (1.00)	0.02 (0.00)	-	-	-
<i>FE</i>	out	0	0.76 (-)	0.06 (-)	0	0.76 (-)	0.06 (-)	-	-	-
	P11-P14	0	0.76 (-)	0.06 (-)	1	0.79 (1.00)	0.05 (0.00)	-	-	-
	P21-P24	7	0.73 (0.71)	0.11 (0.14)	7	0.82 (0.86)	0.02 (0.00)	-	-	-
	P31-P34	1	0.70 (0.00)	0.05 (0.00)	1	0.79 (1.00)	0.05 (0.00)	-	-	-
<i>FP</i>	out	0	0.95 (-)	0.05 (-)	0	0.95 (-)	0.05 (-)	0	0.95 (-)	0.05 (-)
	P11-P14	0	0.95 (-)	0.05 (-)	0	0.95 (-)	0.05 (-)	0	0.95 (-)	0.05 (-)
	P21-P24	8	0.99 (1.00)	0.01 (0.00)	30	0.97 (0.97)	0.03 (0.03)	0	0.95 (-)	0.05 (-)
	P31-P34	2	0.96 (1.00)	0.04 (0.00)	3	0.88 (0.67)	0.12 (0.33)	0	0.95 (-)	0.05 (-)

Table A.3: Input data from all matches except final: Julius Brink – Serves and Attack-Hits

attack strength		<i>normal</i>			<i>hard</i>		
	performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>
Defense	<i>d</i>	20	0.85 (0.85)	0.05 (0.05)	14	0.71 (0.71)	0.21 (0.21)
	<i>d_m</i>	29	0.93 (0.93)	0.00 (0.00)	13	0.46 (0.46)	0.38 (0.38)
Reception	<i>r</i>	34	1.00 (1.00)	0.00 (0.00)	9	0.90 (0.89)	0.10 (0.11)
	<i>r_m</i>	42	0.95 (0.95)	0.02 (0.02)	3	0.97 (1.00)	0.02 (0.00)
Set	<i>s</i>	117	0.99 (0.99)	0.00 (0.00)	-	-	-
	performance	#	<i>block-point</i>	<i>block-ok</i>	<i>block-fault</i>	<i>noblock</i>	
Block	<i>b</i>	5	0.20	0.20	0.20	0.40	

Table A.4: Input data from all matches except final: Julius Brink – Defense, Reception, Set, Block

A.2 Reckermann

target fields performance		#	Q11-Q14		#	Q21-Q24		#	Q31-Q34	
			<i>succ</i>	<i>fault</i>		<i>succ</i>	<i>fault</i>		<i>succ</i>	<i>fault</i>
Serve										
<i>S_F</i>	Po1 - Po4	14	0.93 (0.93)	0.00 (0.00)	8	0.99 (1.00)	0.00 (0.00)	-	-	-
<i>S_J</i>		3	1.00 (1.00)	0.00 (0.00)	4	1.00 (1.00)	0.00 (0.00)	-	-	-
Attack-Hit										
<i>FS</i>	out	0	0.93 (-)	0.00 (-)	0	0.93 (-)	0.00 (-)	-	-	-
	P11-P14	0	0.93 (-)	0.00 (-)	0	0.93 (-)	0.00 (-)	-	-	-
	P21-P24	19	0.89 (0.89)	0.00 (0.00)	8	0.98 (1.00)	0.00 (0.00)	-	-	-
	P31-P34	0	0.93 (-)	0.00 (-)	0	0.93 (-)	0.00 (-)	-	-	-
<i>FE</i>	out	0	0.92 (-)	0.00 (-)	0	0.92 (-)	0.00 (-)	-	-	-
	P11-P14	0	0.92 (-)	0.00 (-)	0	0.92 (-)	0.00 (-)	-	-	-
	P21-P24	0	0.92 (-)	0.00 (-)	0	0.92 (-)	0.00 (-)	-	-	-
	P31-P34	0	0.92 (-)	0.00 (-)	0	0.92 (-)	0.00 (-)	-	-	-
<i>FP</i>	out	0	0.92 (-)	0.00 (-)	0	0.92 (-)	0.00 (-)	0	0.92 (-)	0.00 (-)
	P11-P14	0	0.92 (-)	0.00 (-)	0	0.92 (-)	0.00 (-)	0	0.92 (-)	0.00 (-)
	P21-P24	1	0.83 (0.00)	0.00 (0.00)	10	0.99 (1.00)	0.00 (0.00)	0	0.92 (-)	0.00 (-)
	P31-P34	0	0.92 (-)	0.00 (-)	1	0.92 (1.00)	0.00 (0.00)	0	0.92 (-)	0.00 (-)

Table A.5: Input data from final match: Jonas Reckermann – Serves and Attack-Hits

attack strength		<i>normal</i>				<i>hard</i>		
	performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	
Defense	<i>d</i>	8	0.88 (0.88)	0.00 (0.00)	1	0.89 (1.00)	0.01 (0.00)	
	<i>d_m</i>	6	0.78 (0.83)	0.22 (0.17)	1	0.68 (0.00)	0.31 (1.00)	
Reception	<i>r</i>	7	0.97 (1.00)	0.00 (0.00)	5	0.86 (0.80)	0.00 (0.00)	
	<i>r_m</i>	14	0.93 (0.93)	0.07 (0.07)	7	0.77 (0.71)	0.13 (0.14)	
Set	<i>s</i>	24	0.88 (0.88)	0.04 (0.04)	-	-	-	
	performance	#	<i>block-point</i>	<i>block-ok</i>	<i>block-fault</i>	<i>noblock</i>		
Block	<i>b</i>	63	0.08	0.10	0.16	0.67		

Table A.6: Input data from final match: Jonas Reckermann – Defense, Reception, Set

target fields		Q11-Q14				Q21-Q24				Q31-Q34			
	performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>
Serve													
<i>S_F</i>	Po1 - Po4	25	0.76 (0.76)	0.00 (0.00)	35	0.91 (0.91)	0.06 (0.06)	-	-	-	-	-	-
<i>S_J</i>		38	0.82 (0.82)	0.03 (0.03)	23	0.70 (0.70)	0.30 (0.30)	-	-	-	-	-	-
Attack-Hit													
<i>FS</i>	out	0	0.95 (-)	0.02 (-)	0	0.95 (-)	0.02 (-)	-	-	-	-	-	-
	P11-P14	0	0.95 (-)	0.02 (-)	0	0.95 (-)	0.02 (-)	-	-	-	-	-	-
	P21-P24	49	1.00 (1.00)	0.00 (0.00)	25	0.84 (0.84)	0.08 (0.08)	-	-	-	-	-	-
	P31-P34	9	0.99 (1.00)	0.00 (0.00)	3	0.97 (1.00)	0.02 (0.00)	-	-	-	-	-	-
<i>FE</i>	out	0	0.93 (-)	0.00 (-)	1	0.94 (1.00)	0.00 (0.00)	-	-	-	-	-	-
	P11-P14	0	0.93 (-)	0.00 (-)	0	0.93 (-)	0.00 (-)	-	-	-	-	-	-
	P21-P24	7	0.88 (0.86)	0.00 (0.00)	5	0.96 (1.00)	0.00 (0.00)	-	-	-	-	-	-
	P31-P34	0	0.93 (-)	0.00 (-)	1	0.94 (1.00)	0.00 (0.00)	-	-	-	-	-	-
<i>FP</i>	out	0	0.88 (-)	0.06 (-)	0	0.88 (-)	0.06 (-)	0	0.88 (-)	0.06 (-)	0	0.88 (-)	0.06 (-)
	P11-P14	0	0.88 (-)	0.06 (-)	0	0.88 (-)	0.06 (-)	0	0.88 (-)	0.06 (-)	0	0.88 (-)	0.06 (-)
	P21-P24	3	0.82 (0.67)	0.04 (0.00)	23	0.91 (0.91)	0.04 (0.04)	1	0.89 (1.00)	0.05 (0.00)	1	0.89 (1.00)	0.05 (0.00)
	P31-P34	0	0.88 (-)	0.06 (-)	7	0.87 (0.86)	0.11 (0.14)	0	0.88 (-)	0.06 (-)	0	0.88 (-)	0.06 (-)

Table A.7: Input data from all matches except final: Jonas Reckermann – Serves and Attack-Hits

attack strength		<i>normal</i>			<i>hard</i>		
	performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>
Defense	<i>d</i>	20	0.85 (0.85)	0.10 (0.10)	1	0.74 (0.00)	0.09 (0.00)
	<i>d_m</i>	19	0.84 (0.84)	0.05 (0.05)	0	0.84 (-)	0.05 (-)
Reception	<i>r</i>	27	1.00 (1.00)	0.00 (0.00)	7	0.90 (0.86)	0.10 (0.14)
	<i>r_m</i>	61	0.95 (0.95)	0.02 (0.02)	3	0.97 (1.00)	0.01 (0.00)
Set	<i>s</i>	128	0.98 (0.98)	0.00 (0.00)	-	-	-
	performance	#	<i>block-point</i>	<i>block-ok</i>	<i>block-fault</i>	<i>noblock</i>	
Block	<i>b</i>	200	0.12	0.13	0.14	0.62	

Table A.8: Input data from all matches except final: Jonas Reckermann – Defense, Reception, Set

A.3 Alison

target fields		P11-P14			P21-P24			P31-P34		
performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	
Serve										
<i>S_F</i>	Q01 - Q04	6	0.70 (0.67)	0.00 (0.00)	2	0.80 (1.00)	0.00 (0.00)	-	-	-
<i>S_J</i>	Q01 - Q04	15	0.80 (0.80)	0.07 (0.07)	2	0.86 (1.00)	0.05 (0.00)	-	-	-
Attack-Hit										
<i>FS</i>	out	0	1.00 (-)	0.00 (-)	0	1.00 (-)	0.00 (-)	-	-	-
	Q11-Q14	0	1.00 (-)	0.00 (-)	0	1.00 (-)	0.00 (-)	-	-	-
	Q21-Q24	9	1.00 (1.00)	0.00 (0.00)	1	1.00 (1.00)	0.00 (0.00)	-	-	-
	Q31-Q34	1	1.00 (1.00)	0.00 (0.00)	2	1.00 (1.00)	0.00 (0.00)	-	-	-
<i>FE</i>	out	0	1.00 (-)	0.00 (-)	0	1.00 (-)	0.00 (-)	-	-	-
	Q11-Q14	0	1.00 (-)	0.00 (-)	0	1.00 (-)	0.00 (-)	-	-	-
	Q21-Q24	1	1.00 (1.00)	0.00 (0.00)	1	1.00 (1.00)	0.00 (0.00)	-	-	-
	Q31-Q34	0	1.00 (-)	0.00 (-)	1	1.00 (1.00)	0.00 (0.00)	-	-	-
<i>FP</i>	out	0	1.00 (-)	0.00 (-)	0	1.00 (-)	0.00 (-)	0	1.00 (-)	0.00 (-)
	Q11-Q14	0	1.00 (-)	0.00 (-)	0	1.00 (-)	0.00 (-)	0	1.00 (-)	0.00 (-)
	Q21-Q24	0	1.00 (-)	0.00 (-)	0	1.00 (-)	0.00 (-)	0	1.00 (-)	0.00 (-)
	Q31-Q34	1	1.00 (1.00)	0.00 (0.00)	0	1.00 (-)	0.00 (-)	0	1.00 (-)	0.00 (-)

Table A.9: Input data from final match: Alison Cerutti – Serves and Attack-Hits

attack strength		<i>normal</i>			<i>hard</i>		
performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	
Defense	<i>d</i>	7	0.57 (0.57)	0.29 (0.29)	0	0.56 (-)	0.30 (-)
	<i>d_m</i>	6	0.59 (0.67)	0.35 (0.33)	2	0.41 (0.00)	0.39 (0.50)
Reception	<i>r</i>	6	0.69 (0.67)	0.00 (0.00)	1	0.77 (1.00)	0.00 (0.00)
	<i>r_m</i>	5	0.98 (1.00)	0.00 (0.00)	0	0.91 (-)	0.00 (-)
Set	<i>s</i>	52	1.00 (1.00)	0.00 (0.00)	-	-	-
performance	#	<i>block-point</i>	<i>block-ok</i>	<i>block-fault</i>	<i>noblock</i>		
Block	<i>b</i>	49	0.06	0.16	0.10	0.67	

Table A.10: Input data from final match: Alison Cerutti – Defense, Reception, Set, Block

target fields		P11-P14			P21-P24			P31-P34		
performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	
Serve										
<i>S_F</i>	Q01-Q04	45	0.89 (0.89)	0.00 (0.00)	46	0.96 (0.96)	0.04 (0.04)	-	-	-
<i>S_J</i>	Q01-Q04	37	0.70 (0.70)	0.05 (0.05)	17	0.76 (0.76)	0.24 (0.24)	-	-	-
Attack-Hit										
<i>FS</i>	out	0	0.87 (-)	0.07 (-)	0	0.87 (-)	0.07 (-)	-	-	-
	Q11-Q14	0	0.87 (-)	0.07 (-)	0	0.87 (-)	0.07 (-)	-	-	-
	Q21-Q24	47	0.89 (0.89)	0.04 (0.04)	23	0.83 (0.83)	0.09 (0.09)	-	-	-
	Q31-Q34	8	0.96 (1.00)	0.02 (0.00)	4	0.73 (0.50)	0.23 (0.50)	-	-	-
<i>FE</i>	out	0	0.90 (-)	0.00 (-)	0	0.90 (-)	0.00 (-)	-	-	-
	Q11-Q14	1	0.91 (1.00)	0.00 (0.00)	1	0.91 (1.00)	0.00 (0.00)	-	-	-
	Q21-Q24	4	0.85 (0.75)	0.00 (0.00)	4	0.94 (1.00)	0.00 (0.00)	-	-	-
	Q31-Q34	0	0.90 (-)	0.00 (-)	0	0.90 (-)	0.00 (-)	-	-	-
<i>FP</i>	out	0	0.94 (-)	0.00 (-)	0	0.94 (-)	0.00 (-)	0	0.94 (-)	0.00 (-)
	Q11-Q14	0	0.94 (-)	0.00 (-)	0	0.94 (-)	0.00 (-)	0	0.94 (-)	0.00 (-)
	Q21-Q24	2	0.86 (0.50)	0.00 (0.00)	10	0.99 (1.00)	0.00 (0.00)	0	0.94 (-)	0.00 (-)
	Q31-Q34	1	0.95 (1.00)	0.00 (0.00)	5	0.97 (1.00)	0.00 (0.00)	0	0.94 (-)	0.00 (-)

Table A.11: Input data from all matches except final: Alison Cerutti – Serves and Attack-Hits

attack strength		<i>normal</i>			<i>hard</i>		
	performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>
Defense	<i>d</i>	29	0.83 (0.83)	0.03 (0.03)	8	0.47 (0.38)	0.20 (0.25)
	<i>d_m</i>	17	0.76 (0.76)	0.18 (0.18)	5	0.55 (0.40)	0.42 (0.60)
Reception	<i>r</i>	26	1.00 (1.00)	0.00 (0.00)	5	1.00 (1.00)	0.00 (0.00)
	<i>r_m</i>	36	0.97 (0.97)	0.00 (0.00)	1	0.86 (0.00)	0.12 (1.00)
Set	<i>s</i>	180	0.98 (0.98)	0.00 (0.00)	-	-	-
	performance	#	<i>block-point</i>	<i>block-ok</i>	<i>block-fault</i>	<i>noblock</i>	
Block	<i>b</i>	254	0.13	0.13	0.15	0.59	

Table A.12: Input data from all matches except final: Alison Cerutti – Defense, Reception, Set, Block

A.4 Emanuel

target fields performance		#	P11-P14		#	P21-P24		#	P31-P34	
			<i>succ</i>	<i>fault</i>		<i>succ</i>	<i>fault</i>		<i>succ</i>	<i>fault</i>
Serve										
<i>S_F</i>	Q01-Q04	10	0.90 (0.90)	0.00 (0.00)	9	0.99 (1.00)	0.00 (0.00)	-	-	-
<i>S_J</i>		9	0.89 (0.89)	0.00 (0.00)	2	0.93 (1.00)	0.00 (0.00)	-	-	-
Attack-Hit										
<i>FS</i>	out	0	0.86 (-)	0.05 (-)	0	0.86 (-)	0.05 (-)	-	-	-
	Q11-Q14	0	0.86 (-)	0.05 (-)	0	0.86 (-)	0.05 (-)	-	-	-
	Q21-Q24	21	0.90 (0.90)	0.05 (0.05)	14	0.79 (0.79)	0.07 (0.07)	-	-	-
	Q31-Q34	5	0.83 (0.80)	0.02 (0.00)	4	0.91 (1.00)	0.03 (0.00)	-	-	-
<i>FE</i>	out	0	0.91 (-)	0.03 (-)	0	0.91 (-)	0.03 (-)	-	-	-
	Q11-Q14	0	0.91 (-)	0.03 (-)	0	0.91 (-)	0.03 (-)	-	-	-
	Q21-Q24	2	0.93 (1.00)	0.02 (0.00)	0	0.91 (-)	0.03 (-)	-	-	-
	Q31-Q34	0	0.91 (-)	0.03 (-)	1	0.92 (1.00)	0.02 (0.00)	-	-	-
<i>FP</i>	out	0	0.90 (-)	0.00 (-)	0	0.90 (-)	0.00 (-)	0	0.90 (-)	0.00 (-)
	Q11-Q14	0	0.90 (-)	0.00 (-)	0	0.90 (-)	0.00 (-)	0	0.90 (-)	0.00 (-)
	Q21-Q24	1	0.91 (1.00)	0.00 (0.00)	7	0.87 (0.86)	0.00 (0.00)	0	0.90 (-)	0.00 (-)
	Q31-Q34	0	0.90 (-)	0.00 (-)	2	0.92 (1.00)	0.00 (0.00)	0	0.90 (-)	0.00 (-)

Table A.13: Input data from final match: Emanuel Rego – Serves and Attack-Hits

attack strength		<i>normal</i>			<i>hard</i>		
	performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>
Defense	<i>d</i>	7	0.75 (0.86)	0.20 (0.14)	13	0.38 (0.38)	0.38 (0.38)
	<i>d_m</i>	18	0.33 (0.33)	0.61 (0.61)	7	0.19 (0.14)	0.69 (0.71)
Reception	<i>r</i>	13	0.85 (0.85)	0.00 (0.00)	1	0.71 (0.00)	0.00 (0.00)
	<i>r_m</i>	27	0.85 (0.85)	0.04 (0.04)	0	0.85 (-)	0.04 (-)
Set	<i>s</i>	12	1.00 (1.00)	0.00 (0.00)	-	-	-
	performance	#	<i>block-point</i>	<i>block-ok</i>	<i>block-fault</i>	<i>noblock</i>	
Block	<i>b</i>	2	0.00	0.50	0.00	0.50	

Table A.14: Input data from final match: Emanuel Rego – Defense, Reception, Set, Block

target fields		P11-P14			P21-P24			P31-P34		
performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>	
Serve										
<i>S_F</i>	Q01-Q04	42	0.98 (0.98)	0.02 (0.02)	44	0.84 (0.84)	0.14 (0.14)	-	-	-
<i>S_J</i>	Q01-Q04	32	1.00 (1.00)	0.00 (0.00)	15	0.73 (0.73)	0.20 (0.20)	-	-	-
Attack-Hit										
<i>FS</i>	out	0	0.87 (-)	0.06 (-)	0	0.87 (-)	0.06 (-)	-	-	-
	Q11-Q14	0	0.87 (-)	0.06 (-)	0	0.87 (-)	0.06 (-)	-	-	-
	Q21-Q24	83	0.90 (0.90)	0.02 (0.02)	57	0.81 (0.81)	0.11 (0.11)	-	-	-
	Q31-Q34	12	1.00 (1.00)	0.00 (0.00)	4	0.73 (0.50)	0.22 (0.50)	-	-	-
<i>FE</i>	out	0	1.00 (-)	0.00 (-)	0	1.00 (-)	0.00 (-)	-	-	-
	Q11-Q14	1	1.00 (1.00)	0.00 (0.00)	0	1.00 (-)	0.00 (-)	-	-	-
	Q21-Q24	12	1.00 (1.00)	0.00 (0.00)	5	1.00 (1.00)	0.00 (0.00)	-	-	-
	Q31-Q34	0	1.00 (-)	0.00 (-)	1	1.00 (1.00)	0.00 (0.00)	-	-	-
<i>FP</i>	out	0	0.96 (-)	0.00 (-)	0	0.96 (-)	0.00 (-)	0	0.96 (-)	0.00 (-)
	Q11-Q14	0	0.96 (-)	0.00 (-)	0	0.96 (-)	0.00 (-)	0	0.96 (-)	0.00 (-)
	Q21-Q24	4	0.97 (1.00)	0.00 (0.00)	18	0.94 (0.94)	0.00 (0.00)	1	0.96 (1.00)	0.00 (0.00)
	Q31-Q34	0	0.96 (-)	0.00 (-)	2	0.97 (1.00)	0.00 (0.00)	0	0.96 (-)	0.00 (-)

Table A.15: Input data from all matches except final: Emanuel Rego – Serves and Attack-Hits

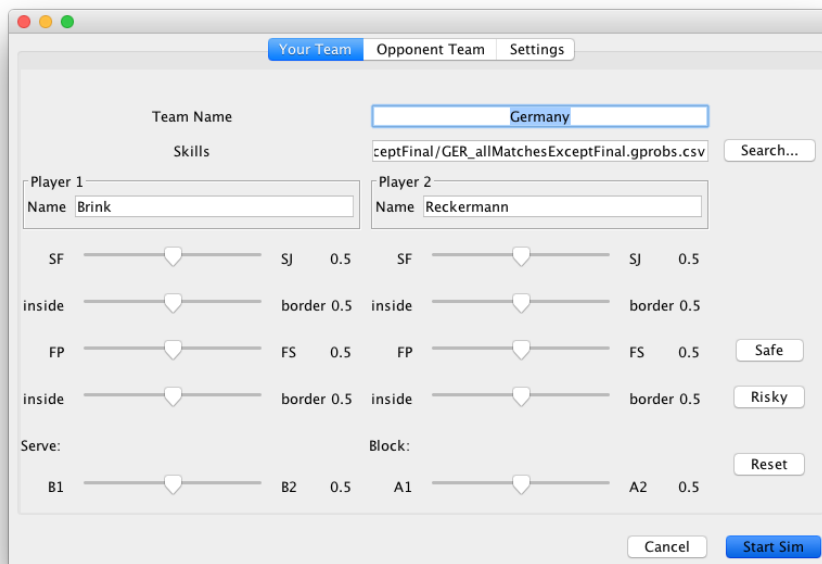
attack strength		<i>normal</i>			<i>hard</i>		
	performance	#	<i>succ</i>	<i>fault</i>	#	<i>succ</i>	<i>fault</i>
Defense	<i>d</i>	22	0.86 (0.86)	0.05 (0.05)	24	0.17 (0.17)	0.50 (0.50)
	<i>d_m</i>	36	0.81 (0.81)	0.17 (0.17)	22	0.41 (0.41)	0.36 (0.36)
Reception	<i>r</i>	69	0.94 (0.94)	0.06 (0.06)	19	0.68 (0.68)	0.05 (0.05)
	<i>r_m</i>	48	0.98 (0.98)	0.00 (0.00)	3	0.99 (1.00)	0.00 (0.00)
Set	<i>s</i>	95	1.00 (1.00)	0.00 (0.00)	-	-	-
	performance	#	<i>block-point</i>	<i>block-ok</i>	<i>block-fault</i>	<i>noblock</i>	
Block	<i>b</i>	11	0.09	0.36	0.09	0.45	

Table A.16: Input data from all matches except final: Emanuel Rego – Defense, Reception, Set, Block

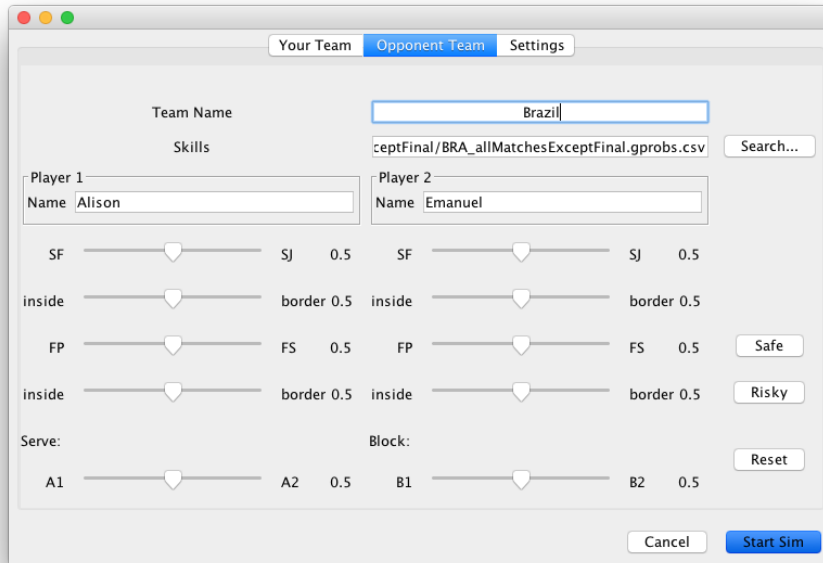
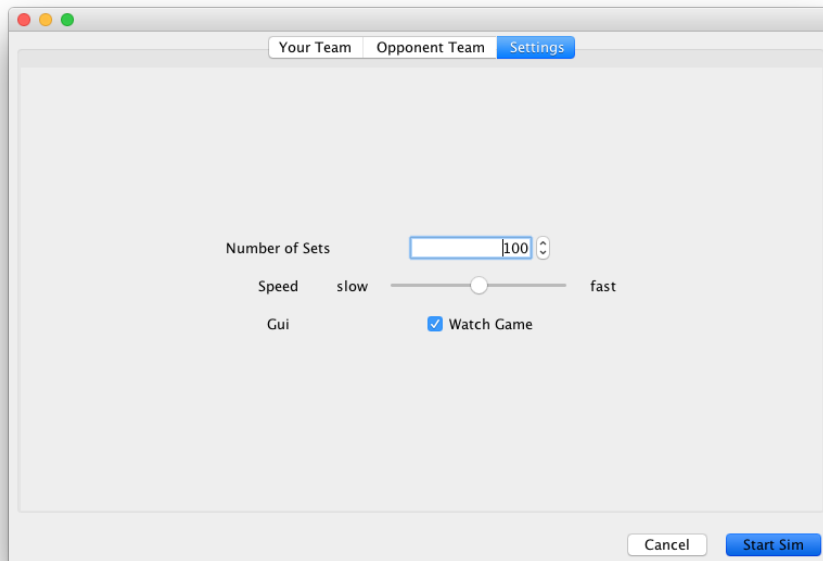
Appendix B

Rally-SSO-MDP: Simulation

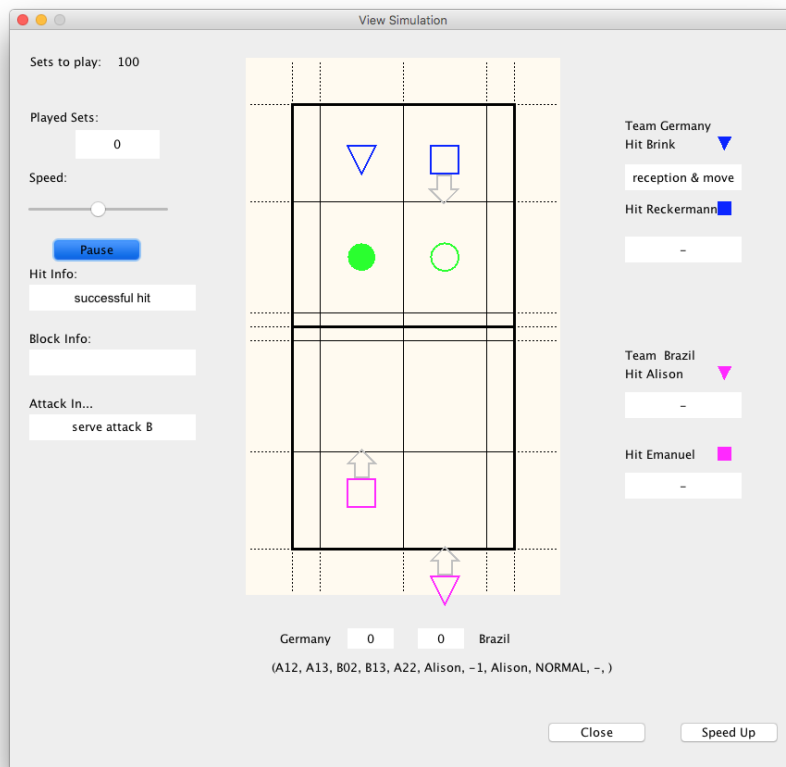
Screenshots of the g-MDP Simulation



(a) Settings of Team P

(b) Settings of Team Q 

(c) General Settings



(d) Simulation Screen

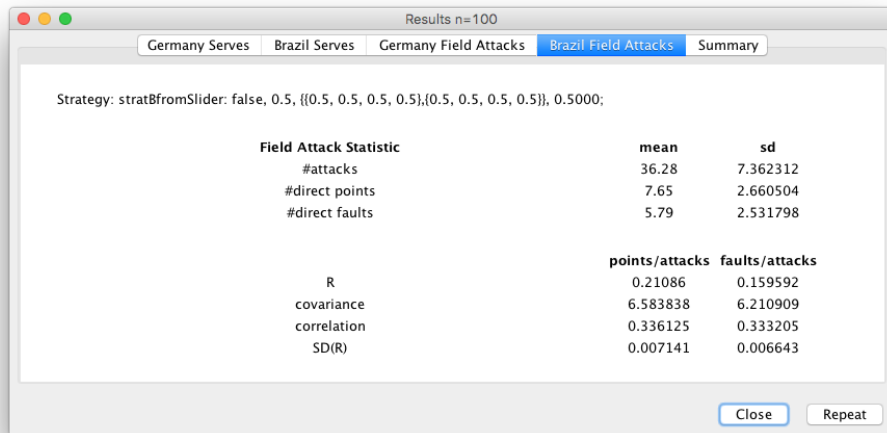
The 'Results n=100' window shows a 'Germany Serves' tab. The strategy is: stratAfromSlider: true, 0.5, {{0.5, 0.5, 0.5, 0.5},{0.5, 0.5, 0.5, 0.5}}, 0.5000. The following table summarizes the serve statistics:

Serve Statistic	mean	sd
#serves	19.99	2.042453
#direct points	0.68	0.85138
#direct faults	3.36	1.678864

	points/serves	faults/serves
R	0.034017	0.168084
covariance	0.431111	-0.006465
correlation	0.247921	-0.001885
SD(R)	0.004186	0.008575

'Close' and 'Repeat' buttons are at the bottom right.

(e) Results: Serves



Results n=100

Germany Serves Brazil Serves Germany Field Attacks **Brazil Field Attacks** Summary

Strategy: stratBfromSlider: false, 0.5, {(0.5, 0.5, 0.5, 0.5)},{(0.5, 0.5, 0.5, 0.5)}, 0.5000;

Field Attack Statistic	mean	sd
#attacks	36.28	7.362312
#direct points	7.65	2.660504
#direct faults	5.79	2.531798

	points/attacks	faults/attacks
R	0.21086	0.159592
covariance	6.583838	6.210909
correlation	0.336125	0.333205
SD(R)	0.007141	0.006643

Close Repeat

(f) Results: Attacks



Results n=100

Germany Serves Brazil Serves Germany Field Attacks Brazil Field Attacks **Summary**

Validation

Won sets by team Germany 75
Set-winning-probability 0.75

Close Repeat

(g) Results: Summary

Figure B.1: Screenshots g-MDP simulation

Appendix C

Rally-SSO-MDP: Basic Decision Rule

C.1 Definition of the Basic Decision Rule

The basic decision rule is used as a default decision rule in the SSO-MDP for a Beach volleyball rally of Section 4.3, which is also called the g-MDP in context of the 2MDP-approach. The basic decision rule specifies a uniform probability distribution over the set of reasonable actions, see Definition 4.3.1. It thereby makes it possible to have some kind of default behavior such that a decision rule regarding binary questions of interest can be characterized by a set of parameters that modify the basic decision rule. In this section, the definition of the basic decision rule is listed in Java-code and supplemented by some explanations.

The basic decision rule is implemented as a Java class and used in the Java simulation of the rally-SSO-MDP respectively the g-MDP. According to the list of binary question, presented in Subsection 4.3.2, the basic decision rule is parametrized by ten parameters, which are

$$\pi = \begin{pmatrix} \pi_b \\ \pi_b \\ \pi_s \end{pmatrix}, \quad \pi_b = \begin{pmatrix} \pi_b^{serve} \\ \pi_b^{field} \end{pmatrix}, \quad \pi_b^{sit} = \begin{pmatrix} \pi_{b,tech}^{sit}(\xi) \\ \pi_{b,field}^{sit}(\xi) \end{pmatrix}, \quad \begin{matrix} sit \in \{serve, field\}, \\ \xi \in \{P_1, P_2\}. \end{matrix}$$

In Listing C.1, the constructor of the parametrized basic decision rule is presented. The class is named *Strategy* since, in a sport context, it is the commonly used term. Besides the specification of the parameters a central field for each player is set depending whether the strategy is a strategy of team P or of team Q .¹ The name of the strategy is just a meaningful string that helps to identify a specified parametrization.

```
1 public Strategy(String name, boolean isTeamA, double pi_b,  
   double [][] pi_h, double pi_o1){  
2   this.name = name;  
3   this.isTeamA = isTeamA;  
4   this.pi_b = pi_b;  
5   this.pi_h = pi_h;  
6   this.pi_o1 = pi_o1;  
7
```

¹The description variables or functions in the the presented pseudo-code uses the letters A respectively B instead of P and Q to name the participating teams.

```

8   if (isTeamA){
9       centralFieldP1 = Field.A22;
10      centralFieldP2 = Field.A23;
11  }else{
12      centralFieldP1 = Field.B22;
13      centralFieldP2 = Field.B23;
14  }
15 }

```

Listing C.1: Constructor parametrized rally-SSO-MDP strategy

In this implementation, the basic decision rule specifies the reasonable action sets R_s , depending on the category of state the s , see Table 4.5. In the remainder of this section, we specify these reasonable action sets for each state category and whether the team is in possession of the ball or not. In each decision rule, it is specified whether it is a decision rule of team P . Therefore, it is possible to determine from the current position of the ball whether the team who chooses its next team action is in possession of the ball or not.

As already mentioned in Subsection 4.3.2, each of the binary decision questions will partition the reasonable action sets in two sets. The basic decision rule can be regained from the parametrization presented in Table 4.8.

Assume in the following, a decision rule that is a decision rule of team P . This assumption helps to avoid writing “the team who is using this decision rule” and to avoid annoying case distinctions. All reasonable action sets can analogously be constructed for a decision rule of team Q since the rally SSO-MDP is built with a symmetric view on team P and team Q .

C.1.1 Serving States

If the current state s is a serving state of team P , i.e., $s \in S_p^{serve}$, it is first determined whether player 1 or player 2 is hitting. As all serving states of team P are explicitly listed in the set S_p^{serve} , also two subsets can be specified to distinguish between serving states of player 1 and player 2. This is done in a natural manner and used to determine the hitting player. In the next step, a hit and a movement are determined for the hitting player according to the parametrization of the decision rule. The function `chooseServeAfterStrat` determines the hitting technique and the target field. It is listed and explained in Listing C.3. If the chosen hit is a jump serve, the server makes one step forward to the net else he moves towards a field on the side of the field that is not covered by the other player. Thereby the function `moveTowardsField` returns a one-step movement in the direction towards the specified field. And `getCentralFieldOfOtherHalf(field)` is a function that returns the central field that is on the court half not containing `field`. The central fields are specified in the strategy class as $P23$ and $P22$ for team P . The non-hitting player stays in his field and does nothing. Finally, the returned team action is constructed. Depending on whether player 1 or player 2 is the hitting player, the player actions must be the first or the second player action in the constructor of the team action.

```

1 public TeamAction servingRule(State s, double r) {
2     // which player is hitting
3     int playerNumber;
4     if (Arrays.asList(State.START_STATES_A1).contains(s)){
5         playerNumber = 1;

```



```

6   }else{
7       playerNumber = 2;
8   }
9
10  // determine hit
11  Hit h = Hit.chooseServeAfterStrat(isTeamA,
12      pi_h[playerNumber-1][0], pi_h[playerNumber-1][1], pi_o1);
13
14  Move moveServer;
15  if (h.isHitWithJump()){
16      moveServer = Move.m_f;
17  }else{
18      if (Arrays.asList(Field.OUT_A).contains(fp1)){
19          moveServer =
20              fp1.moveTowardsField(getCentralFieldOfOtherHalf(fp2));
21      }else{
22          moveServer =
23              fp2.moveTowardsField(getCentralFieldOfOtherHalf(fp1));
24      }
25  }
26
27  if (playerNumber == 1)
28      return new TeamAction(new PlayerAction(h,moveServer), new
29          PlayerAction(Hit.nohit, Move.stay));
30  else
31      return new TeamAction(new PlayerAction(Hit.nohit, Move.stay),
32          new PlayerAction(h,moveServer));
33 }

```

Listing C.2: Reasonable action sets of serving states

The function `chooseServeAfterStrat`, listed in Listing C.3, determines a serving technique and a target field according to the probability distribution specified by $\pi_{b,tech}^{serve}(\varrho)$ and $\pi_{b,field}^{serve}(\varrho)$. Also the parameter π_s is taken into account to determine the court half of the target field. The function `MainSim.countRandomCalls()` is a function that returns a random double value in $[0, 1)$ computed by the Java Function `nextDouble()` of the `Random` class. The drawing of random numbers is encapsulated in a function for debugging reasons. For instance, in `MainSim.countRandomCalls()` the total number of random calls is counted and only a single `Random` object is used such that the specification of a certain seed is possible. So, after the first draw of a random number, it is evaluated whether this number is smaller than $\pi_{b,tech}^{serve}(\varrho)$ or not. If it is smaller the a *SJ* is chosen else a *SF*. This leads to a probability distribution where a *SJ* is chosen with probability $\pi_{b,tech}^{serve}(\varrho)$ as desired. Similarly, the target field is determined. First, it is evaluated from another drawing of a random number whether the target field is a border field or not, and second, whether it is a field of the right or the left half of the opponent's court. By definition of the serving states, player 1 of the opponent team is placed in field *Q12* and player 2 in field *Q13*. Observe, that only the side edge fields without the fields at the net are considered as border fields; these are precisely 4 fields on each court half and thus as many as the non-border fields that are no fields at the net.

```

1 public static Hit chooseServeAfterStrat(boolean isTeamA, double
    pi_serve_tech, double pi_serve_field, double pi_o1){
2     HitTechnique tech;
3     Field f;
4
5     if (MainSim.countRandomCalls() < pi_serve_tech){
6         tech = HitTechnique.S_J;
7     }else{
8         tech = HitTechnique.S_F;
9     }
10
11    if (MainSim.countRandomCalls() < pi_serve_field){
12        if(MainSim.countRandomCalls() < pi_o1){
13            f = Field.chooseRandomField(new Field[]{Field.B11,
14                Field.B21});
15        }else{
16            f = Field.chooseRandomField(new Field[]{Field.B14,
17                Field.B24});
18        }
19    }else{
20        if(MainSim.countRandomCalls() < pi_o1){
21            f = Field.chooseRandomField(new Field[]{Field.B12,
22                Field.B22});
23        }else{
24            f = Field.chooseRandomField(new Field[]{Field.B13,
25                Field.B23});
26        }
27    }
28    return new Hit(tech, f);
29 }

```

Listing C.3: Choose Service

If the current state s is a serving state of the opponent team, i.e., $s \in S_Q^{serve}$ the team does nothing. The team action `doNothing` is defined by two player actions that consists of the hit *no hit* and the move *stay*.

```

1 public TeamAction otherTeamServingRule(State s, double r) {
2     return TeamAction.doNothing;
3 }

```

Listing C.4: Reasonable action sets of other team serving

C.1.2 Reception States

The general idea of the reception rule, listed in Listing C.5, is to let one player make a reception towards the central field of the other half of the court and let the other player move towards this target field. If a player is in the same field as the ball, the technique r is used. If a player is only in a neighbor field of the ball a reception with a move, r_m , is used. As a reception with a move is, in general, a more difficult hit compared to a reception without a movement, it is first tested whether a player is at the

position of the ball and only if both players do not fulfill that, it is determined whether a player is in a neighbor-field of the ball. Since both players may be in the same field or both players are in a neighbor-field of the ball, it is selected with an equal probability which player's position is tested first. The function `ballIsInsideMyCourtSide()` determines whether the ball is inside the court on the team's side and prevents from receiving a ball that is in an outside field. If no player can receive the ball, the `doNothing` team action is returned. As in the next step, the rally will be completed if no real hit is performed, a movement wouldn't have any influence in any case.

```

1 public TeamAction receptionRule(State s, double r) {
2     /* randomize which player is tested first */
3     int order = 1;
4     if (r < 0.5){
5         order = 1;
6     }else{
7         order = 2;
8     }
9
10    for (int i = 0; i < 2; i++){
11        switch (order){
12            case 1:
13                if (fP1 == fBall && ballIsInsideMyCourtSide()){
14                    Field targetField = getCentralFieldOfOtherHalf(fP1);
15                    Hit h = new Hit(HitTechnique.r, targetField);
16                    return new TeamAction(new PlayerAction(h, Move.stay),
17                                           new PlayerAction(Hit.nohit,
18                                                             fP2.moveTowardsField(targetField)));
19                }
20                order = 2;
21                break;
22            case 2:
23                if (fP2 == fBall && ballIsInsideMyCourtSide()){
24                    Field targetField = getCentralFieldOfOtherHalf(fP2);
25                    Hit h = new Hit(HitTechnique.r, targetField);
26                    return new TeamAction(new PlayerAction(Hit.nohit,
27                                                             fP1.moveTowardsField(targetField)), new
28                                           PlayerAction(h, Move.stay));
29                }
30                order = 1;
31                break;
32            }
33        }
34    }
35
36    for (int i = 0; i < 2; i++){
37        switch (order){
38            case 1:
39                if (fP1.isNeighbour(fBall) && ballIsInsideMyCourtSide()){
40                    Field targetField = getCentralFieldOfOtherHalf(fP1);
41                    Hit h = new Hit(HitTechnique.r_m, targetField);

```

```

37         return new TeamAction(new PlayerAction(h, Move.stay),
38                                new PlayerAction(Hit.nohit,
39                                                  fP2.moveTowardsField(targetField)));
38     }
39     order = 2;
40     break;
41 case 2:
42     if (fP2.isNeighbour(fBall) && ballIsInsideMyCourtSide()){
43         Field targetField = getCentralFieldOfOtherHalf(fP2);
44         Hit h = new Hit(HitTechnique.r_m, targetField);
45         return new TeamAction(new PlayerAction(Hit.nohit,
46                                                  fP1.moveTowardsField(targetField)), new
47                                PlayerAction(h, Move.stay));
46     }
47     order = 1;
48     break;
49 }
50 }
51
52 return TeamAction.doNothing;
53 }

```

Listing C.5: Reasonable action sets of receiving states

If the current state is a receiving state of the other team, both players start positioning themselves on the field to defend the next attack hit, see Listing C.6. The function `determineBlockingPlayer` is listed below and determines a designated blocking player according to the parameter π_b . The designated blocking player moves – if he is not already there – forward to the net, while the non-blocking player moves to the central field on the other court half of the blocking player. The decision who is the designated blocking player of the next attack is part of the team action and as specified in the system dynamic in the next state. This inclusion in the state means that this decision can be accessed at the next time step.

```

1 public TeamAction otherTeamReceptionRule(State s, double r) {
2     // non-blocking player moves towards the central fields
3     Move moveP1 =
4         fP1.moveTowardsField(getCentralFieldOfOtherHalf(fP2));
5     Move moveP2 =
6         fP2.moveTowardsField(getCentralFieldOfOtherHalf(fP1));
7     TeamAction result = new TeamAction(new PlayerAction(Hit.nohit,
8                                                         moveP1), new PlayerAction(Hit.nohit, moveP2));
9
10    Player blockingPlayer = determineBlockingPlayer(r);
11    result.setDesignatedBlockingPlayer(blockingPlayer);
12
13    // blocker moves forward except he is already at the net
14    Move moveBlocker;
15    if (blockingPlayer.isAtTheNet()){
16        moveBlocker = Move.stay;
17    }else{

```

```

15     moveBlocker = Move.m_f;
16 }
17 // overwrite player action of blocking player
18 result.setPlayerActionOfPlayer(blockingPlayer, new
    PlayerAction(Hit.nohit, moveBlocker));
19
20 return result;
21 }

```

Listing C.6: Reasonable action sets of other team receiving

```

1 private Player determineBlockingPlayer(double r){
2     if (r < pi_b){
3         return Game.getInstance().getTeamA().getPlayer1();
4     }else{
5         return Game.getInstance().getTeamA().getPlayer2();
6     }
7 }

```

Listing C.7: Determine the blocking player

C.I.3 Defense States

The defending rule is similar to the reception rule. The only differences are that a defense technique instead of a receiving technique is used. Furthermore, only a player who is not in a field directly at the net should perform that defense hit. As in the receiving states, if no defense action is possible by both players the team action do nothing is returned.

```

1 public TeamAction defendingRule(State s, double r) {
2     /* randomize which player is tested first */
3     int order = 1;
4     if (r < 0.5){
5         order = 1;
6     }else{
7         order = 2;
8     }
9
10    for (int i = 0; i < 2; i++){
11        switch (order){
12            case 1:
13                if (fP1 == fBall&&
14                    (!Arrays.asList(Field.AT_THE_NET_A).contains(fP1) &&
15                     ballIsInsideMyCourtSide())){
16                    Field targetField = getCentralFieldOfOtherHalf(fP1);
17                    Hit h = new Hit(HitTechnique.d, targetField);
18                    return new TeamAction(new PlayerAction(h, Move.stay),
19                                           new PlayerAction(Hit.nohit,
20                                                                fP2.moveTowardsField(targetField)));
21                }
22            case 2:
23                if (fP2 == fBall&&
24                    (!Arrays.asList(Field.AT_THE_NET_B).contains(fP2) &&
25                     ballIsInsideMyCourtSide())){
26                    Field targetField = getCentralFieldOfOtherHalf(fP2);
27                    Hit h = new Hit(HitTechnique.d, targetField);
28                    return new TeamAction(new PlayerAction(h, Move.stay),
29                                           new PlayerAction(Hit.nohit,
30                                                                fP1.moveTowardsField(targetField)));
31                }
32            default:
33                return new TeamAction(new PlayerAction(Hit.nohit, fP1),
34                                       new PlayerAction(Hit.nohit, fP2));
35        }
36    }
37 }

```

```

18     order=2;
19     break;
20     case 2:
21     if (fP2 == fBall&&
22         (!Arrays.asList(Field.AT_THE_NET_A).contains(fP2) &&
23          ballIsInsideMyCourtSide())){
24         Field targetField = getCentralFieldOfOtherHalf(fP2);
25         Hit h = new Hit(HitTechnique.d, targetField);
26         return new TeamAction(new PlayerAction(Hit.nohit,
27         fP1.moveTowardsField(targetField)), new
28         PlayerAction(h, Move.stay));
29     }
30     order=1;
31     break;
32 }
33 }
34
35 for (int i = 0; i < 2; i++){
36     switch (order){
37     case 1:
38     if (fP1.isNeighbour(fBall) &&
39         (!Arrays.asList(Field.AT_THE_NET_A).contains(fP1) &&
40          ballIsInsideMyCourtSide())){
41         Field targetField = getCentralFieldOfOtherHalf(fP1);
42         Hit h = new Hit(HitTechnique.d_m, targetField);
43         return new TeamAction(new PlayerAction(h, Move.stay),
44         new PlayerAction(Hit.nohit,
45         fP2.moveTowardsField(targetField)));
46     }
47     order=2;
48     break;
49     case 2:
50     if (fP2.isNeighbour(fBall) &&
51         (!Arrays.asList(Field.AT_THE_NET_A).contains(fP2) &&
52          ballIsInsideMyCourtSide())){
53         Field targetField = getCentralFieldOfOtherHalf(fP2);
54         Hit h = new Hit(HitTechnique.d_m, targetField);
55         return new TeamAction(new PlayerAction(Hit.nohit,
56         fP1.moveTowardsField(targetField)), new
57         PlayerAction(h, Move.stay));
58     }
59     order=1;
60     break;
61     }
62 }
63 return TeamAction.doNothing;
64 }

```

Listing C.8: Reasonable action sets of defending states

If the other team is defending, exactly the same team action is returned as when the other team is receiving the ball after a service.

```

1 public TeamAction otherTeamDefendingRule(State s, double r) {
2     // non-blocking player moves towards the central fields
3     Move moveP1 = fP1.moveTowardsField(centralFieldP1);
4     Move moveP2 = fP2.moveTowardsField(centralFieldP2);
5     TeamAction result = new TeamAction(new PlayerAction(Hit.nohit,
6         moveP1), new PlayerAction(Hit.nohit, moveP2));
7
8     Player blockingPlayer = determineBlockingPlayer(r);
9     result.setDesignatedBlockingPlayer(blockingPlayer);
10
11    // blocker moves forward except he is already at the net
12    Move moveBlocker;
13    if (blockingPlayer.isAtTheNet()){
14        moveBlocker = Move.stay;
15    }else{
16        moveBlocker = Move.m_f;
17    }
18    // overwrite player action of blocking player
19    result.setPlayerActionOfPlayer(blockingPlayer, new
20        PlayerAction(Hit.nohit, moveBlocker));
21
22    return result;
23 }

```

Listing C.9: Reasonable action sets of other team defending

C.I.4 Setting States

The `settingRule` specified in Listing C.10 defines the team action for setting states of team P . Again the order in which the players are tested is randomized. The first tested player who is in the same position as the ball, and who has not last touched the ball will perform the set. As for the setting technique, only a neighbor-field of player's current position is allowed, it is tested whether the central field of the other court half belongs to the neighbor-fields of the own field. If that's the case, a set towards the central field of the other court half is made, and the non-hitting player moves towards that target field. If that's not the case a fall-back to the reception rule is made. A reception has fewer requirements than a setting, so it may be the case that no setting is possible but a reception can be performed. A fall-back to a reception and not to defense is made since a regain of the ball after a failed setting is easier than defending an attack hit and therefore more similar to a reception.

```

1 public TeamAction settingRule(State s, double r) {
2     /* randomize which player is tested first */
3     int order = 1;
4     if (r < 0.5){
5         order = 1;
6     }else{
7         order = 2;

```

```

8     }
9
10    for (int i = 0; i < 2; i++){
11        switch (order){
12            case 1:
13                if(fP1 == fBall && s.getLastContact() !=
14                    Game.getInstance().getTeamA().getPlayer1()){
15                    Field targetField = getCentralFieldOfOtherHalf(fP1);
16                    if (targetField.isNeighbourOrOwnField(fP1)){
17                        Hit hitP1 = new Hit(HitTechnique.s,targetField);
18                        Field moveTargetField;
19                        moveTargetField = targetField;
20                        return new TeamAction(new PlayerAction(hitP1,
21                            Move.stay), new PlayerAction(Hit.nohit,
22                                fP2.moveTowardsField(moveTargetField)));
23                    }
24                }
25                order = 2;
26                break;
27            case 2:
28                if(fP2 == fBall && s.getLastContact() !=
29                    Game.getInstance().getTeamA().getPlayer2()){
30                    Field targetField = getCentralFieldOfOtherHalf(fP2);
31                    if (targetField.isNeighbourOrOwnField(fP2)){
32                        Hit hitP2 = new Hit(HitTechnique.s,targetField);
33                        Field moveTargetField;
34                        moveTargetField = targetField;
35                        return new TeamAction(new PlayerAction(Hit.nohit,
36                            fP1.moveTowardsField(moveTargetField)), new
37                                PlayerAction(hitP2, Move.stay));
38                    }
39                }
40                order = 1;
41                break;
42            }
43        }
44    }
45    return receptionRule(s,r);
46 }

```

Listing C.10: Reasonable action sets of setting states

If the other team does the setting prior to their next attack, the positioning on the court for defending the next attack hit is done. First, it is estimated which player will possibly perform the next attack hit. The function `findColOfPossiblyAttackingPlayer` returns an estimate of the column from which the next attack hit will be performed. The function `findColOfPossiblyAttackingPlayer` is described and listed below in more detail. Having an estimate of the column of the next attack hit, the blocking player moves towards a field at the net of this column. If the blocking player is already at the net, he may move left or right to be in the column of the estimated next attack. If he is in the correct

column and at the net, he stays there. The non-blocking player positions himself in the central field on the court half that is not covered by the blocking player. Finally, depending on whether player 1 or player 2 is blocking, the team action is constructed from both player actions.

```

1 public TeamAction otherTeamSettingRule(State s, double r) {
2     // determine column of the player who probably will do the
3     // attack hit in the next step
4     int col_possiblyAttackingCol =
5         findColOfPossiblyAttackingPlayer(s);
6     PlayerAction blockPlayerAction;
7     Field blockerField;
8     Player designatedBlocker;
9     if (s.getBlocker() == null){
10        // this could be the case if the ball was directly before
11        // blocked
12        designatedBlocker = determineBlockingPlayer(r);
13    }else{
14        designatedBlocker = s.getBlocker();
15    }
16
17    if (designatedBlocker ==
18        Game.getInstance().getTeamA().getPlayer1())
19        blockerField = fP1;
20    else
21        blockerField = fP2;
22
23    if(col_possiblyAttackingCol < blockerField.getColumn()){
24        if (Arrays.asList(Field.AT_THE_NET_A).contains(blockerField)){
25            blockPlayerAction = new PlayerAction(Hit.nohit, Move.m_r);
26        }else{
27            // blocker one step forward-right
28            blockPlayerAction = new PlayerAction(Hit.nohit, Move.m_fr);
29        }
30    }else if (col_possiblyAttackingCol > blockerField.getColumn()){
31        if (Arrays.asList(Field.AT_THE_NET_A).contains(blockerField)){
32            blockPlayerAction = new PlayerAction(Hit.nohit, Move.m_l);
33        }else{
34            // blocker one step forward-left
35            blockPlayerAction = new PlayerAction(Hit.nohit, Move.m_lf);
36        }
37    }else{
38        if (Arrays.asList(Field.AT_THE_NET_A).contains(blockerField)){
39            blockPlayerAction = new PlayerAction(Hit.nohit, Move.stay);
40        }else{
41            // blocker one step forward
42            blockPlayerAction = new PlayerAction(Hit.nohit, Move.m_f);
43        }
44    }
45    // non blocking player moves towards the central field of the

```

```

    court half which is not covered by the block
42 PlayerAction nonblockingPlayerAction;
43 Field nonBlockingPlayerField;
44 if (designatedBlocker ==
    Game.getInstance().getTeamA().getPlayer1()){
45     nonBlockingPlayerField = fP2;
46 }else{
47     nonBlockingPlayerField = fP1;
48 }
49
50 if (col_possiblyAttackingCol <=
    nonBlockingPlayerField.getColumn()){
51     nonblockingPlayerAction = new PlayerAction(Hit.nohit,
    nonBlockingPlayerField.moveTowardsField(centralFieldP2));
52 }else{
53     nonblockingPlayerAction = new PlayerAction(Hit.nohit,
    nonBlockingPlayerField.moveTowardsField(centralFieldP1));
54 }
55
56 if (designatedBlocker ==
    Game.getInstance().getTeamA().getPlayer1()){
57     return new TeamAction(blockPlayerAction,
    nonblockingPlayerAction, designatedBlocker);
58 }else{
59     return new TeamAction(nonblockingPlayerAction,
    blockPlayerAction, designatedBlocker);
60 }
61 }
62 }

```

Listing C.11: Reasonable action sets of other team setting

The function `findColOfPossiblyAttackingPlayer` is listed in Listing C.12. It suggests, depending on the current state, the player who will perform a hit in the next step. This suggestion is made by determining whether one of the players is currently in the same position as the ball. If that's the case, it is assumed that this player will perform a hit in the current state and the other player will perform a hit in the next state. Therefore, the column of the player who is not in possession of the ball is returned. The column of a field is defined as the second index. So, for example, the field *P23* is in column 3. The columns correspond to the rows in Figure 4.6 because we horizontally draw the court. If none of the players is at the same position as the ball, it can not be suggested who will perform a hit in the current state. In that case, the column of the position of the ball is returned.

```

1 private int findColOfPossiblyAttackingPlayer(State s){
2     // find column of the player who is not in possession of the
3     // ball and who will probably perform the attack hit
4     int col_possiblyAttackingCol;
5     if (s.getFieldBall() == s.getFieldB1())
6         col_possiblyAttackingCol = s.getFieldB2().getColumn();
7     else if (s.getFieldBall() == s.getFieldB2())
8         col_possiblyAttackingCol = s.getFieldB1().getColumn();

```

```

8   else
9       // if the attacking player can not be guessed return the
        column of the ball
10      col_possiblyAttackingCol = s.getFieldBall().getColumn();
11
12      return col_possiblyAttackingCol;
13  }

```

Listing C.12: Find Column of Possibly Attacking Player

C.1.5 Attacking States

In an attacking state of team P , it is determined whether one of the players can make a proper attack (smash or planned shot), or whether at least an emergency shot is possible. If none of those options is possible, the `doNothing` team action is returned, as shown at the end in Listing C.13. A fall-back to a defense or a receiving action makes no sense since those techniques only allow to choose a target field on the own courtside, see Table 4.6. Since, according to the classification of an attacking state, the counter is already 2, the ball must cross the net at this point, else the three-hits-rule is violated in the next step. If one player is at the position of the ball, he chooses the technique and target field according to the parametrization of the decision rule. This is done in the function `chooseFieldAttackAfterStrat` described below. The non-hitting player does nothing. If no player is in the correct position, an emergency shot can still be allowed if a player is in a neighbor-field of the ball. In this case, a random shot from all shots available is selected.

```

1  public TeamAction attackingRule(State s, double r) {
2      int playerNumber = 0;
3
4      // determine attacking player
5      if(fP1==fBall && s.getLastContact() !=
        Game.getInstance().getTeamA().getPlayer1()){
6          playerNumber=1;
7      }
8      if(fP2==fBall && s.getLastContact() !=
        Game.getInstance().getTeamA().getPlayer2()){
9          playerNumber=2;
10     }
11
12     Hit h;
13     if (playerNumber != 0){
14         // determine hit according to parametrization
15         h = Hit.chooseFieldAttackAfterStrat(isTeamA,
            pi_h[playerNumber-1][2], pi_h[playerNumber-1][3]);
16         if (playerNumber == 1){
17             return new TeamAction(new PlayerAction(h, Move.stay),
                PlayerAction.doNothing);
18         } else if (playerNumber == 2){
19             return new TeamAction(PlayerAction.doNothing, new
                PlayerAction(h, Move.stay));

```

```

20     }
21 }else{
22     // no player at the ball -> emergency shot
23     // ball in neighbour-field
24     if(fBall.isNeighbour(fp1) && s.getLastContact() !=
        Game.getInstance().getTeamA().getPlayer1()){
25         h = Hit.chooseRandomHit(Hit.SHOT_TEAM_A);
26         return new TeamAction(new PlayerAction(h, Move.stay),
            PlayerAction.doNothing);
27     }
28     if(fBall.isNeighbour(fp2) && s.getLastContact() !=
        Game.getInstance().getTeamA().getPlayer2()){
29         h = Hit.chooseRandomHit(Hit.SHOT_TEAM_A);
30         return new TeamAction(PlayerAction.doNothing, new
            PlayerAction(h, Move.stay));
31     }
32 }
33
34 return TeamAction.doNothing;
35 }

```

Listing C.13: Reasonable action sets of attacking states

The function `chooseFieldAttackAfterStrat`, listed in Listing C.14, determines an attack hit according to the probability distribution specified by $\pi_{b,tech}^{field}(\varrho)$ and $\pi_{b,field}^{field}(\varrho)$. First, the used technique is determined. If the drawn random number is smaller than $\pi_{b,tech}^{field}(\varrho)$, a smash *FS* is chosen, else a planned shot *FP*. Afterwards the target field is determined according to $\pi_{b,tech}^{field}(\varrho)$. Since a planned shot can have a field directly behind the net as a target field, the sets of border fields and non-border fields from which a random target field is selected, differs for the *FS* and the *FP*.

```

1 public static Hit chooseFieldAttackAfterStrat(boolean isTeamA,
        double pi_field_tech, double pi_field_field){
2     HitTechnique tech;
3     Field f;
4
5     if (MainSim.countRandomCalls() < pi_field_tech){
6         tech = HitTechnique.F_SM;
7     }else{
8         tech = HitTechnique.F_P;
9     }
10
11    if (MainSim.countRandomCalls() < pi_field_field){
12        if (tech!=HitTechnique.F_SM){
13            // border fields including fields at the net
14            f = Field.chooseRandomField(new Field[]{Field.B11,
                Field.B21, Field.B31, Field.B14, Field.B24, Field.B34});
15        }else{
16            // border fields excluding fields at the net

```

```

17     f = Field.chooseRandomField(new Field[]{Field.B11,
18         Field.B21, Field.B14, Field.B24});
19 }
20 }else{
21     if (tech!=HitTechnique.F_SM){
22         // non-border fields including fields at the net
23         f = Field.chooseRandomField(new Field[]{Field.B12,
24             Field.B22, Field.B32, Field.B13, Field.B23, Field.B33});
25     }else{
26         // non-border fields excluding fields at the net
27         f = Field.chooseRandomField(new Field[]{Field.B12,
28             Field.B22, Field.B13, Field.B23});
29     }
30 }
31 }
32 return new Hit(tech, f);
33 }

```

Listing C.14: Choose attack hit

Finally, we have the decision rule of team P for states, where team Q makes an attack hit, see Listing C.15. Independent from the specified designated blocking player of the current state, the blocking player is determined as the player who is at the net. If the player is in the same column as the ball, he performs a blocking action. If he is not in the correct column, he moves one step backward to be able to make a defense action. The non-blocking player moves, if he is not already there, towards the central field on the other court half. Depending on whether player 1 or player 2 is the blocking player, a team action is constructed and returned from these player actions.

```

1 public TeamAction otherTeamAttackingRule(State s, double r) {
2     Player blocker = s.getBlocker();
3     PlayerAction blockingPlayerAction;
4     Field blockingPlayerField;
5     PlayerAction nonBlockingPlayerAction;
6     Field nonBlockerField;
7     // determine field of blocking player from position
8     if (s.getBlocker() ==
9         Game.getInstance().getTeamA().getPlayer1()){
10        blockingPlayerField = fP1;
11        nonBlockerField = fP2;
12    }
13    else {
14        blockingPlayerField = fP2;
15        nonBlockerField = fP1;
16    }
17    // blocking player is in the correct column?
18    if (s.getFieldBall().isInSameColumn(blockingPlayerField)
19        && blocker.isAtTheNet()){
20        // yes -> perform block
21        blockingPlayerAction = new PlayerAction(Hit.nohit, Move.b);
22    }else{

```

```
23     // no -> move backwards
24     blockingPlayerAction = new PlayerAction(Hit.nohit, Move.m_b);
25 }
26
27 // non-blocking player moves to central field on other court
28     half
29     if (blockingPlayerField.getColumn() <= 2){
30         nonBlockingPlayerAction = new PlayerAction(Hit.nohit,
31             nonBlockerField.moveTowardsField(centralFieldP2));
32     }else{
33         nonBlockingPlayerAction = new PlayerAction(Hit.nohit,
34             nonBlockerField.moveTowardsField(centralFieldP1));
35     }
36
37     if (blocker == Game.getInstance().getTeamA().getPlayer1()){
38         return new TeamAction(blockingPlayerAction,
39             nonBlockingPlayerAction);
40     }else{
41         return new TeamAction(nonBlockingPlayerAction,
42             blockingPlayerAction);
43     }
44 }
```

Listing C.15: Reasonable action sets of other team attacking

Appendix D

Set-SSO-MDP: Strategy Estimates of Team Q

Based on pre-final matches						
strategy	#	q^{serve}	\bar{q}^{serve}	#	q^{field}	\bar{q}^{field}
<i>risky-risky</i>	19	5%	32%	78	65%	21%
<i>risky-safe</i>	19	5%	32%	27	37%	0%
<i>safe-risky</i>	146	1%	7%	78	65%	21%
<i>safe-safe</i>	146	1%	7%	27	37%	0%

(a) pre-final setting

Based final match						
strategy	#	q^{serve}	\bar{q}^{serve}	#	q^{field}	\bar{q}^{field}
<i>risky-risky</i>	6	17%	33%	32	69%	19%
<i>risky-safe</i>	6	17%	33%	7	14%	0%
<i>safe-risky</i>	22	0%	9%	32	69%	19%
<i>safe-safe</i>	22	0%	9%	7	14%	0%

(b) post-final setting

Table D.1: Direct estimation of s-MDP probabilities for team Q

Appendix E

Data and Software

List of Figures

2.1	Example MAXPROB problem that is no Bertsekas-SSP	34
2.2	Hierarchy of MDP classes	35
2.3	Conserving decision rule is not sufficient (Puterman, 2005, Ex. 7.2.3)	41
2.4	Policy Iteration may terminate with a suboptimal policy	47
2.5	Many feasible dual solutions to a decision rule	51
2.6	LP formulation for NEG models: non convex feasible regions	62
2.7	Illustration of non convex feasible regions of Example 2.7	63
3.1	SSO-MDP Example	78
3.2	MDP classes hierarchy SSO-MDPs included	82
3.3	SSO-MDP Example	92
3.4	Solution of SSO-MDP Example	93
3.5	Flow network example of an SSO-MDP	97
3.6	Flow generated by random process of SSO-MDP in time-extended network	102
3.7	Flow network example of an SSO-MDP	123
3.8	Illustration of Algorithm 6 for $e \in E : e \in \delta_{\neg}(C)$ with $end(e) \in N_S$	140
3.9	Illustration of Algorithm 6 for $e \in E : e \in \delta_{\neg}(C)$ with $end(e) \in N_A$	141
3.10	Modeling a sprint in soccer as constraint SSO-MDP	151
3.11	SSO-MG Example	153
4.1	Set-SSO-MDP	160
4.2	Tie-game	161
4.3	Refined regular SSO-MDP	165
4.4	Aggregated regular SSO-MDP	166
4.5	Aggregated Tie-Game	166
4.6	Court grid	175
4.7	Beach Volleyball Tracker – User Interface	194
4.8	Beach Volleyball Tracker – Example Data	196
4.9	Aggregation Scheme	199
5.1	Modeling Decision – Degrees of Detail	212
5.2	Two Scale Approach	214
5.3	Skill-Strategy Score Card: <i>safe</i> versus <i>risky</i> play for varying smash skills (<i>FS</i>)	228

5.4	Skill-Strategy Score Card: <i>safe</i> versus <i>risky</i> play for varying shot skills (FP)	229
5.5	Skill-Strategy Score Card: <i>safe</i> versus <i>risky</i> play for varying jump serve skills (S_J)	230
5.6	Skill-Strategy Score Card: <i>safe</i> versus <i>risky</i> play for varying float serve skills (S_F) and $p_{\text{fault},g}(\text{field}, S_F)$	231
B.1	Screenshots g-MDP simulation	254

List of Tables

4.1	An SSO-MDP modeling a beach volleyball set (set-SSO-MDP)	163
4.2	Match statistics of the final match at the Olympic Beach Volleyball Tournament 2012 in London (Database, 2018)	170
4.3	Estimated transition probabilities from the final match	172
4.4	Comparing strategies against the final strategy of Brazil	173
4.5	State Categories	176
4.6	Hit specification for team P	178
4.7	Move specification for g belonging to team P	179
4.8	Parameters settings for regaining basic decision rule	193
4.9	Internet statistics of the Olympic Beach Volleyball Tournament 2012 in London – Final match included	196
4.10	Attack Statistic from collected g-data – final match included	197
4.11	Serve Statistic from collected g-data – final match included	197
4.12	Input data from all matches: Julius Brink – Serves and Attack-Hits	201
4.13	Input data from all matches: Julius Brink – Defense, Reception, Set, Block	201
4.14	Input data from all matches: Jonas Reckermann – Serves and Attack-Hits	202
4.15	Input data from all matches: Jonas Reckermann – Defense, Reception, Set	202
4.16	Input data from all matches: Alison Cerutti – Serves and Attack-Hits	203
4.17	Input data from all matches: Alison Cerutti – Defense, Reception, Set, Block	203
4.18	Input data from all matches: Emanuel Rego – Serves and Attack-Hits	204
4.19	Input data from all matches: Emanuel Rego – Defense, Reception, Set, Block	204
4.20	Generated state space by basic decision rule	206
5.1	Comparison rough versus detailed SSO-MDP	213
5.2	Meaningful s-g-implementation $\tau_{\mathcal{A}}(s^j, a^j) \mapsto \pi$	219
5.3	Validation of simulated s-MDP transition probabilities	221
5.4	Estimation of s-MDP probabilities from g-MDP simulation	222
5.5	Direct estimation of s-MDP probabilities	225
5.6	Comparison between 2-MDP approach and direct approach	226
5.7	Winning probabilities of Germany regarding different strategy combinations in the pre-final setting.	233
5.8	Winning probabilities of Germany regarding different strategy combinations in the post-final setting.	234

A.1	Input data from final match: Julius Brink – Serves and Attack-Hits	241
A.2	Input data from final match: Julius Brink – Defense, Reception, Set, Block	242
A.3	Input data from all matches except final: Julius Brink – Serves and Attack-Hits	242
A.4	Input data from all matches except final: Julius Brink – Defense, Reception, Set, Block	243
A.5	Input data from final match: Jonas Reckermann – Serves and Attack-Hits	243
A.6	Input data from final match: Jonas Reckermann – Defense, Reception, Set	244
A.7	Input data from all matches except final: Jonas Reckermann – Serves and Attack-Hits	244
A.8	Input data from all matches except final: Jonas Reckermann – Defense, Reception, Set	245
A.9	Input data from final match: Alison Cerutti – Serves and Attack-Hits	245
A.10	Input data from final match: Alison Cerutti – Defense, Reception, Set, Block	246
A.11	Input data from all matches except final: Alison Cerutti – Serves and Attack-Hits	246
A.12	Input data from all matches except final: Alison Cerutti – Defense, Reception, Set, Block	247
A.13	Input data from final match: Emanuel Rego – Serves and Attack-Hits	247
A.14	Input data from final match: Emanuel Rego – Defense, Reception, Set, Block	248
A.15	Input data from all matches except final: Emanuel Rego – Serves and Attack-Hits	248
A.16	Input data from all matches except final: Emanuel Rego – Defense, Reception, Set, Block	249
D.1	Direct estimation of s-MDP probabilities for team Q	271

Acronyms

Bertsekas-SSP MDP	Bertsekas Shortest Path MDP
GSSP MDP	General Stochastic Shortest Path MDP
MAXPROB MDP	Maximum Probability MDP
NEG MDP	Negative MDP
POSB MDP	Positive Bounded MDP
SSO-MDP	Sport-Strategy Optimization MDP
SSO-MG	Sport-Strategy Optimization MG
SSP MDP	Stochastic Shortest Path MDP
SSPADE	Stochastic Shortest Path MDP with avoidable Dead Ends
SSPUDE	Stochastic Shortest Path MDP with unavoidable Dead Ends
g-MDP	Gameplay MDP
s-MDP	Strategic MDP
2-MDP approach	Two-MDP Approach
MDP	Markov Decision Problem
MG	Markov Game

Symbols

A_s	Set of actions available in state s
A'_s	Set of state-action pairs available in state s
T	Decision points
d_s	Decision rule in state s
$q_s(a)$	Probability of choosing action a in state s under decision rule d
B	Dynamic programming operator
N	Horizon
m	Number of actions
n	Number of states
π	Policy
π^*	Optimal policy
d^∞	Stationary policy
$2^{\mathcal{A}}$	Powerset of set \mathcal{A}
$\mathcal{P}(\mathcal{A})$	Probability distribution over set \mathcal{A}
\mathbb{P}	Probability measure
$r_t(s, a)$	Expected reward from action a in state s at time t
$r_t(s, a, s')$	Reward from transition from state s under actions a to state s' at time t
$r_N(s)$	Terminal reward in state s
r_d	Expected reward vector from decision rule d
Y	Random variable for action of Markov process
Z	Random variable for node in flow network
X	Random variable for state of Markov process
\mathcal{A}	Set of actions

D	Set of all decision rules
D^{MD}	Set of all deterministic Markovian decision rules
D^{MR}	Set of all randomized Markovian decision rules
Π^{MD}	Set of all deterministic Markovian policies
G	Set of goal states
L	Set of losing states
Π	Set of all policies
Π^{MR}	Set of all randomized Markovian policies
S	Set of states
V	Set of value functions
W	Set of winning states
A'	Set of state-actions pairs
l	Losing state
s_1	Starting state
w	Winning state
$p_t(s' s, a)$	Transition probability from state s under actions a to state s' at time t
P_d	Transition matrix under decision rule d
v	Value function of an MDP
$v^*(s)$	Value of MDP starting in state s (infinite horizon)
$v_N^*(s)$	Value of MDP starting in state s (finite horizon)
$g^\pi(s)$	Expected average reward of policy π (infinite horizon)
$v_\lambda^\pi(s)$	Expected discounted reward of policy π (infinite horizon)
$v^\pi(s)$	Expected total reward of policy π (infinite horizon)
$v_N^\pi(s)$	Expected total reward of policy π in (finite Horizon)

Bibliography

- Adler, Ilan (2010). “On the Equivalence of Linear Programming Problems and Zero-Sum Games.” In: *Optimization Online* 1.June, pp. 1–7.
- Ahmann, Jörg (2001). *Vergleichende Struktur- und Sachanalyse beim internationalen Beach-Volleyball der Herren hinsichtlich Regeländerungen zur Saison 2001*. Tech. rep. Sonder-Lehrgang A-Trainer des DVV.
- Ahuja, Ravindra, Thomas Magnanti, and James Orlin (1993). *Network Flows: Theory, algorithms, and applications*, p. 846. ISBN: 0-13-617549-x. DOI: 10.1016/0166-218X(94)90171-6.
- Anbarci, Nejat, Ching-Jen Sun, and M. Utku Ünver (2015). *Designing Fair Tiebreak Mechanisms: The Case of Penalty Shootouts*. Tech. rep., pp. 1–52.
- Anderson, Ross et al. (2007). “Algorithmic Game Theory.” In: *Algorithmic Game Theory*, p. 754. ISSN: 00010782. DOI: 10.1145/1785414.1785439. arXiv: 0907.4385.
- Bellman, Richard (1957a). *A Markovian decision process*. DOI: 10.1007/BF02935461.
- Bellman, Richard (1957b). *Dynamic programming*. Princeton, NJ, USA: Princeton University Press, p. 339.
- Bertsekas, Dimitri P. (2001). *Dynamic Programming and Optimal Control. Volume II*. 2nd ed. Belmont, Massachusetts: Athena Scientific, p. 303. ISBN: 1-886529-27-2.
- Bertsekas, Dimitri P. (2005). *Dynamic Programming and Optimal Control. Volume I*. 3rd ed. Belmont, Massachusetts: Athena Scientific, p. 543. ISBN: 1-886529-26-4.
- Borodin, Allan and Ran El-Yaniv (2005). *Online computation and competitive analysis*. Cambridge University Press.
- Bukiet, Bruce, Elliotte Rusty Harold, and José Luis Palacios (1997). “A Markov chain approach to baseball.” In: *Operations Research* 45.1, pp. 14–23.
- Buscà, Bernat et al. (2012). “The influence of serve characteristics on performance in men’s and women’s high-standard beach volleyball.” In: *Journal of Sports Sciences* 30.3, pp. 269–276. DOI: 10.1080/02640414.2011.635309.

- Cañal-Bruland, Rouwen, Merel Mooren, and Geert J. Savelsbergh (2011). "Differentiating experts' anticipatory skills in beach volleyball." In: *Research Quarterly for Exercise and Sport* 82.4, pp. 667–674. ISSN: 21683824. DOI: 10.1080/02701367.2011.10599803.
- Chan, Timothy C. Y. and Raghav Singal (2016). "A Markov Decision Process-based handicap system for tennis." In: *Journal of Quantitative Analysis in Sports* 12.4, pp. 179–189. ISSN: 1559-0410. DOI: 10.1515/jqas-2016-0057.
- Clarke, Stephen R. and John M. Norman (1998). "Dynamic programming in cricket: Protecting the weaker batsman." In: *Asia Pacific Journal of Operational Research* 15.1. ISSN: 02175959.
- Clarke, Stephen R. and John M. Norman (1999). "To run or not?: Some dynamic programming models in cricket." In: *Journal of the Operational Research Society* 50.5, pp. 536–545. ISSN: 0160-5682. DOI: 10.1057/palgrave.jors.2600705.
- Clarke, Stephen R. and John M. Norman (2012). "Optimal challenges in tennis." In: *Journal of the Operational Research Society* 63.12, pp. 1765–1772. ISSN: 0160-5682. DOI: 10.1057/jors.2011.147.
- Cortell-Tormo, Juan M. et al. (2011). "Analysis of Movement Patterns by Elite Male Players of Beach Volleyball." In: *Perceptual and Motor Skills* 112.1, pp. 21–28. ISSN: 0031-5125. DOI: 10.2466/05.27.PMS.112.1.21-28.
- Dantzig, George B. (1951). "A proof of the equivalence of the programming problem and the game problem." In: *Activity analysis of production and allocation* 13, pp. 330–338.
- Dantzig, George B. (1963). *Linear Programming and Extensions*.
- Database, Beachvolleball (2018). *Beach Volleyball Database*. URL: <http://www.bvbinfo.com/Tournament.asp?ID=2594&Process=Matches> (visited on 05/22/2018).
- D'Epenoux, Francois (1963). "A Probabilistic Production and Inventory Problem." In: *Management Science* 10.1, pp. 98–108. DOI: 10.1287/mnsc.10.1.98.
- Fédération Internationale De Volleyball (2012a). *Player Ranking by Skill*. URL: <http://www.fivb.org/en/olympics/london2012/PDF/B6-MLON2012.pdf> (visited on 05/28/2018).
- Fédération Internationale De Volleyball (2012b). *Team Ranking by Skill*. Tech. rep., p. 1. URL: <http://www.fivb.org/en/olympics/london2012/PDF/B5-MLON2012.pdf>.
- Fédération Internationale De Volleyball (2016). *Official Volleyball Rules 2017-2020*. URL: http://www.fivb.org/en/Refereeing-Rules/documents/FIVB-Volleyball{_}Rules2013-EN{_}v2{_}20130422.pdf.
- Ferrante, Marco and Giovanni Fonseca (2014). "On the winning probabilities and mean durations of volleyball." In: *Journal of Quantitative Analysis in Sports* 10.2, pp. 1–8. ISSN: 1559-0410. DOI: 10.1515/jqas-2013-0098.
- Florence, Lindsay W. et al. (2008). "Skill Evaluation in Women's Volleyball." In: *Journal of Quantitative Analysis in Sports* 4.2, p. 14. ISSN: 1559-0410. DOI: 10.2202/1559-0410.1105.

- Giatsis, George (2003). "The effect of changing the rules on score fluctuation and match duration in the FIVB women's beach volleyball." In: *International Journal of Performance Analysis in Sport* 3.1, pp. 57–64. DOI: 10.1080/24748668.2003.11868275.
- Giatsis, George and Zahariadis Panagiotis (2008). "Statistical Analysis of Men's FIVB Beach Volleyball Team Performance." In: *International Journal of Performance Analysis in Sport* 8.1, pp. 31–43. DOI: 10.1080/24748668.2008.11868420.
- Gomez, Gabriel et al. (2014). "Tracking of Ball and players in beach volleyball videos." In: *PLOS ONE* 9.11, pp. 1–19. ISSN: 19326203. DOI: 10.1371/journal.pone.0111730.
- Guillot, Matthieu and Gautier Stauffer (2017). "The Stochastic Shortest Path Problem: A polyhedral combinatorics perspective."
- Hansen, Eric A. (2017). "Error bounds for stochastic shortest path problems." In: *Mathematical Methods of Operations Research* 86.1, pp. 1–27. ISSN: 1432-5217. DOI: 10.1007/s00186-017-0581-5.
- Heiner, Matthew, Gilbert W. Fellingham, and Camille Thomas (2014). "Skill importance in women's soccer." In: *Journal of Quantitative Analysis in Sports* 0.0, pp. 287–302. ISSN: 1559-0410. DOI: 10.1515/jqas-2013-0119.
- Hirotsu, Nobuyoshi and Mike Wright (2002). "Using a Markov process model of an association football match to determine the optimal timing of substitution and tactical decisions." In: *Journal of the Operational Research Society* 53.1, pp. 88–96. ISSN: 0160-5682. DOI: 10.1057/palgrave/jors/2601254.
- Hirotsu, Nobuyoshi and Mike Wright (2003a). "A Markov Chain Approach To Optimal Pinch Hitting Strategies in a Designated Hitter Rule Baseball Game." In: *Journal of the Operations Research Society of Japan* 46.3, pp. 353–371.
- Hirotsu, Nobuyoshi and Mike Wright (2003b). "Determining the best strategy for changing the configuration of a football team." In: *Journal of the Operational Research Society* 54.8, pp. 878–887. ISSN: 0160-5682. DOI: 10.1057/palgrave.jors.2601591.
- Hoffmeister, Susanne and Jörg Rambau (2017a). "Sport Strategy Optimization in Beach Volleyball - How to bound direct point probabilities dependent on individual skills." In: *MathSport Proceedings*, pp. 184–193.
- Hoffmeister, Susanne and Jörg Rambau (2017b). "Strategy Optimization in Sports – A Two-Scale Approach via Markov Decision Problems." URL: http://www.wm.uni-bayreuth.de/de/download/xcf2d3wd41kj2/preprint{_}sso{_}bv.pdf.
- Hoffmeister [formerly Börner], Susanne (2014). "Markovsche Entscheidungsprobleme für Sportspiele." Masterarbeit. University of Bayreuth.
- Howard, Ronald A. (1960). *Dynamic Programming and Markov Processes*. Published jointly by the Technology Press of the Massachusetts Institute of Technology and.
- Johnson, Donald B. (1975). "Finding All the Elementary Circuits of a Directed Graph." In: *SIAM Journal on Computing* 4.1, pp. 77–84. ISSN: 0097-5397. DOI: 10.1137/0204007.

- Jungnickel, Dieter (2008). *Graph, Networks, and Algorithms*. Vol. 5. 7. Springer-Verlag Berlin Heidelberg, p. 781. ISBN: 978-3-642-32277-8. DOI: 10.1007/978-3-642-32278-5.
- Karmarkar, Narendra (1984). "A new polynomial-time algorithm for linear programming." In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pp. 302–311.
- Khachiyan, Leonid G. (1980). "Polynomial algorithms in linear programming." In: *USSR Computational Mathematics and Mathematical Physics* 20.1, pp. 53–72.
- Kira, Akifumi et al. (2015). "A dynamic programming algorithm for optimizing baseball strategies." In: *MI Preprint Series*. Vol. 10, p. 30.
- Klee, Victor and Georg J. Minty (1972). *How Good is the Simplex Algorithm?* Tech. rep., pp. 159–175.
- Koch, Christina and Markus Tilp (2009a). "Analysis of Beach Volleyball Action Sequences of Female Top Athletes." In: *Journal of Human Sport & Exercise* 4.3, pp. 272–283. DOI: 10.4100/jhse.
- Koch, Christina and Markus Tilp (2009b). "Beach Volleyball Techniques and Tactics: A Comparison of Male and Female Playing Characteristics." In: *Kinesiology* 41, pp. 52–59.
- Kolobov, Andrey and Mausam (2012). *Planning with Markov Decision Processes*. Morgan & Claypool Publishers. ISBN: 9781608458868.
- Kolobov, Andrey, Mausam, and S. Weld (2012). "Stochastic Shortest Path MDPs with Dead Ends." In: *ICAPs workshop: Heuristics and Search for Domain Independent Planning*. Sao Paulo, pp. 78–86. URL: <http://icaps12.icaps-conference.org/workshops/hsdip2012-proceedings.pdf>.
- Kolobov, Andrey et al. (2011). "Heuristic search for generalized stochastic shortest path MDPs." In: *Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling*, pp. 130–137.
- Krause, Jakob (2017). "Is more data always better? Optimal data usage in non-stationary systems."
- Künzell, Stefan et al. (2014). "Effectiveness of the call in beach volleyball attacking play." In: *Journal of Human Kinetics* 44.1, pp. 183–191. ISSN: 18997562. DOI: 10.2478/hukin-2014-0124.
- Littman, Michael L. (1994). "Markov games as a framework for multi-agent reinforcement learning." In: *Proceedings of the International Conference on Machine Learning*. Vol. 157. 1, pp. 157–163. ISBN: 1-55860-335-2. DOI: 10.1.1.48.8623.
- Littman, Michael L., Thomas L. Dean, and Leslie P. Kaelbling (1995). "On the complexity of solving Markov decision problems." In: *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 394–402. ISBN: 1-55860-385-9. DOI: 10.1007/11871842. arXiv: 1302.4971.
- Liu, Tianbiao and Andreas Hohmann (2013). "Applying the Markov Chain theory to Analyze the Attacking Actions between FC Barcelona and Manchester United in the European Champions League final." In: *International Journal of Sports Science and Engineering* 07.02, pp. 79–86.

- McGarry, Tim and Ian M. Franks (1994). "A stochastic approach to predicting competition squash match-play." In: *Journal of Sports Sciences* 12.6, pp. 573–584. ISSN: 1466447X. DOI: 10.1080/02640419408732208.
- Miskin, Michelle A., Gilbert W. Fellingham, and Lindsay W. Florence (2010). "Skill Importance in Women's Volleyball Skill Importance in Women's Volleyball." In: *Journal of Quantitative Analysis in Sports* 6.2. ISSN: 1559-0410. DOI: 10.2202/1559-0410.1234.
- Mitchell, Tom M. (2017). "Estimating Probabilities." In: *Machine Learning*. Chap. 2, pp. 1–11. ISBN: 0070428077.
- Nadimpalli, Vamsi K. and John J. Hasenbein (2013). "When to challenge a call in tennis: A Markov decision process approach." In: *Journal of Quantitative Analysis in Sports* 9.3, pp. 229–238.
- Nash, John (1951). "Non-Cooperative Games." In: *Annals of Mathematics* 54.2, pp. 286–295.
- Natali, Simone et al. (2017). "Physical and technical demands of elite beach volleyball according to playing position and gender." In: *The Journal of sports medicine and physical fitness* November, pp. 1–18. ISSN: 1827-1928. DOI: 10.23736/S0022-4707.17.07972-5.
- Newton, Paul K. and Kamran Aslam (2009). "Monte Carlo Tennis: A Stochastic Markov Chain Model." In: *Journal of Quantitative Analysis in Sports* 5.3. ISSN: 1559-0410. DOI: 10.2202/1559-0410.1169.
- Norman, John M. (1985). "Dynamic programming in tennis - when to use a fast serve." In: *Journal Operational Research Society* 36.1, pp. 75–77.
- OddsShark (2017). *odds shark*. URL: <http://www.odds shark.com/sports-betting/which-sport-do-betting-underdogs-win-most-often> (visited on 11/03/2017).
- Oracle. *Java SE 6 HotSpot[tm] Virtual Machine Garbage Collection Tuning*. URL: http://www.oracle.com/technetwork/java/javase/gc-tuning-6-140523.html{\#}par{_}gc.oom (visited on 08/24/2017).
- Osband, Ian et al. (2017). "Deep exploration via randomized value functions."
- Palao, José Manuel, David Valades, and Enrique Ortega (2012). "Match duration and number of rallies in men's and women's 2000-2010 FIVB world tour beach volleyball." In: *Journal of Human Kinetics* 34.1, pp. 99–104. ISSN: 16405544. DOI: 10.2478/v10078-012-0068-7.
- Papadimitriou, Christos H. (1994). "On the complexity of the parity argument and other inefficient proofs of existence." In: *Journal of Computer and System Sciences* 48.3, pp. 498–532. ISSN: 10902724. DOI: 10.1016/S0022-0000(05)80063-7.
- Papadimitriou, Christos H. and John N. Tsitsiklis (1987). "The complexity of Markov Decision Processes." In: *Operations Research* 12.3, pp. 441–450.
- Pfeiffer, Mark (2005). *Leistungsdiagnostik im Nachwuchstraining*. Sport und Buch Strauß.
- Pfeiffer, Mark, Hui Zhang, and Andreas Hohmann (2010). "A Markov chain model of elite table tennis competition." In: *International Journal of Sports Science and Coaching* 5.2, pp. 205–222.

- Puterman, Martin L. (2005). *Markov Decision Processes Discrete Stochastic Dynamic Programming*. New York: John Wiley & Sons, p. 649. ISBN: 0-471-72782-2.
- Routley, Kurt and Oliver Schulte (2015). "A Markov Game Model for Valuing Player Actions in Ice Hockey." In: *Uncertainty in Artificial Intelligence (UAI)*, pp. 782–791.
- Russo, Daniel J. et al. (2018). "A tutorial on thompson sampling." In: *Foundations and Trends in Machine Learning* 11.1, pp. 1–96.
- Sanner, Scott (2010). *Relational dynamic influence diagram language (rddl): Language description*. Tech. rep.
- Sarkar, Sumit (2018). "Paradox of crosses in association football (soccer) - A game-theoretic explanation." In: *Journal of Quantitative Analysis in Sports* 14.1, pp. 25–36. ISSN: 15590410. DOI: 10.1515/jqas-2017-0073.
- Schulte, Oliver et al. (2017). "A Markov Game model for valuing actions, locations, and team performance in ice hockey." In: *Data Mining and Knowledge Discovery*. ISSN: 1384-5810. DOI: 10.1007/s10618-017-0496-z.
- Shapley, Lloyd S. (1953). "Stochastic Games." In: *Proceedings of the National Academy of Sciences* 39.10, pp. 1095–1100. ISSN: 0027-8424. DOI: 10.1073/pnas.39.10.1095.
- Shirley, Kenny (2007). "A Markov model for basketball." In: *New England Symposium for Statistics in Sports*, pp. 82–82.
- Štrumbelj, Erik and Petar Vračar (2012). "Simulating a basketball match with a homogeneous Markov model and forecasting the outcome." In: *International Journal of Forecasting* 28.2, pp. 532–542.
- Terroba, Antonio et al. (2013). "Finding Optimal Strategies in Tennis From Video Sequences." In: *International Journal of Pattern Recognition and Artificial Intelligence* 27.06, pp. 1–31. ISSN: 0218-0014. DOI: 10.1142/S0218001413550100.
- Tseng, Paul (1990). "Solving H-horizon, stationary Markov decision problems in time proportional to $\log(H)$." In: *Operations Research Letters* 9.5, pp. 287–297. ISSN: 01676377. DOI: 10.1016/0167-6377(90)90022-W.
- Turocy, Theodore L. (2008). "In Search of the "Last-Ups" Advantage in Baseball: A Game-Theoretic Approach." In: *Journal of Quantitative Analysis in Sports* 4.2. ISSN: 1559-0410. DOI: 10.2202/1559-0410.1104.
- Von Neumann, John (1928). "Zur Theorie der Gesellschaftsspiele." In: *Mathematische Annalen* 100.1, pp. 295–320. ISSN: 00255831. DOI: 10.1007/BF01448847.
- Walker, Mark, John Wooders, and Rabah Amir (2011). "Equilibrium play in matches: Binary Markov games." In: *Games and Economic Behavior* 71.2, pp. 487–502. ISSN: 08998256. DOI: 10.1016/j.geb.2010.04.011.

- Wright, Mike and Nobuyoshi Hirotsu (2003). "The professional foul in football: Tactics and deterrents." In: *Journal of the Operational Research Society* 54.3, pp. 213–221. ISSN: 0160-5682. DOI: 10.1057/palgrave.jors.2601506.
- Ye, Yinyu (2011). "The Simplex and Policy-Iteration Methods Are Strongly Polynomial for the Markov Decision Problem with a Fixed Discount Rate." In: *Mathematics of Operations Research* 36.4, pp. 593–603. ISSN: 0364-765X. DOI: doi : 10 . 1287 / moor . 1110 . 0516. arXiv: arXiv : 1208 . 5083v1.

Own Publications

Hoffmeister, Susanne and Jörg Rambau (2017). “Sport Strategy Optimization in Beach Volleyball - How to bound direct point probabilities dependent on individual skills.” In: *MathSport Proceedings*, pp. 184–193.

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die von mir angegebenen Quellen und Hilfsmittel verwendet habe.

Weiterhin erkläre ich, dass ich die Hilfe von gewerblichen Promotionsberatern bzw. -vermittlern oder ähnlichen Dienstleistern weder bisher in Anspruch genommen habe, noch künftig in Anspruch nehmen werde.

Zusätzlich erkläre ich hiermit, dass ich keinerlei frühere Promotionsversuche unternommen habe.

Bayreuth, den