Masterarbeit

am
Lehrstuhl für Kommunikationstechnik
Professor Dr. -Ing. habil. Matthias Wolff

Brandenburgische Technische Universität Cottbus - Senftenberg
Fakultät für Maschinenbau, Elektrotechnik und
Wirtschaftsingenieurwesen
Institut für Elektronik und Informationstechnik

Duale Optimierung von verknüpften MDPs

Eingereicht von:

Name : Yu Dong
Matr.-Nr.: 3643873
Anschrift: Universitätsstraße 4
03046 Cottbus

Betreut von:

M.Sc. Werner Meyer

Cottbus, 14.08.2018

THEMENANGEBOT FÜR EINE

MASTERARBEIT

Lehrstuhl
Kommunikationstechnik

Thema:

Duale Optimierung von verknüpften MDPs

Beschreibung:

Ein wesentlicher Bestandteil technischer kognitiver Systeme ist die Verhaltenssteuerung. Diese stützt sich auf eine Strategie, welche in einer vorangegangenen Lernphase erworben wurde. Mit einer Bewertung der Kosten möglicher Handlungen kann eine optimale Strategie gefunden werden, wobei als Optimalitätskriterium eine Kostenminimierung verwendet wird. Um ein bestimmtes Ziel zu erreichen, erzeugt ein klassisches System also eine Sequenz von Entscheidungen, welche eine Koordination der Handlungen zur Folge hat (Planung).

Kognitive Systeme müssen jedoch unter Unsicherheit entscheiden und planen können. Daher müssen zur Modellierung solcher Prozesse stochastische Verfahren verwendet werden (**M**arkov-**D**ecision-**P**rocess)

Das Ziel der Masterarbeit ist die Modellierung der Interaktion von zwei Agenten in einer Labyrinth-Umgebung. Die Agenten sind jeweils mit einem MDP-Verfahren ausgestattet und beeinflussen sich gegenseitig. Es sollen verschiedene Möglichkeiten der Verknüpfung untersucht und mit einander verglichen werden. Das Verhalten bei verknüpften Prozessen soll grafisch dargestellt werden.

Teilaufgaben:
- Erweiterung des Matlab-Codes um weiteren Agenten
- Realisierung einer grafischen Ausgabe

Fähigkeiten:
- Matlab-Kenntnisse erwünscht

Betreuer:

M. Sc. Werner Meyer

T: 0355 69-4353

E: Werner.Meyer@b-tu.de

Selbstständigkeitserklärung

Ich erkläre, dass ich die vorliegende Master-Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt hat. Die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind ausnahmslos als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Cottbus, 14.08.2018

_____

Danksagung

An dieser Stelle möchte ich mich zuerst bei meinem Betreuer Herr M. Sc. Werner Meyer für Ihre gute Betreuung bedanken. Er ermutigt mich sehr oft. Wenn ich eine Frage auftrat, antwortete er immer sehr gut und sehr deutlich. Während unserer Diskussion, erhielt ich immer viel Information, dadurch ich eine neue Idee oder eine neue Methode bekommen könnte.

Ich möchte noch bei Prof. Dr.-Ing. habil. Matthias Wolff und Dr.-Ing. Ronald Römer bedanken. Während des Zwischenvortrags gaben Sie mir viel Hilfe. Wir diskutierten auch einige interessante Themen. Die Mitarbeiter aus der Fakultät Kommunikationstechnik sind sehr freundlich. Ich mache viel Spaß während der Arbeitszeit.

Zum Schluss möchte ich noch bei meinen Eltern bedanken. Sie finanzieren mir mein Studium in Deutschland. Wenn ich in Schwierigkeit bin, geben sie mir den Mut und Vertrauen, alles gut zu machen.

Kurzfassung

In dieser Arbeit wird ein duales MDP-Modell implementiert, um das Verhältnis der verknüpften MDP Verfahren zu analysieren. Shannon´s Maus „Theseus" wird als ein Simulationsmodell verwendet. Eine Katze wird als zweiter Agent im Labyrinth hinzugefügt. Die Maus und die Katze können sich in der Simulation gegenseitig beeinflussen. Ein neu aufgebautes Labyrinth wird verwendet, damit die Wände des Labyrinths keine leeren Plätze besetzen. Die drei im Modell angewendeten MDP Verfahren sind Wert-Iteration, Q-Lernen und TD-Learning. Das Problem des Maus-Katze-Spiels ist ein dynamisches Entscheidungsproblem. Das Ziel der Maus ist es, den kürzesten und sichersten Weg zum Käse zu finden. Die angewendeten MDP Verfahren der beiden Agenten laufen parallel.

Zuerst wird das duale MDP-Modell in der sicheren Umgebung simuliert. Die Belohnungsdifferenzen und die gefundene Strategie der Agenten werden grafisch dargestellt. Durch die Veränderung der vier Belohnungen zeigt sich, dass sich die gefundenen Strategien beeinflussen können. Das duale MDP-Modell kann auch unter Unsicherheit eine Strategie finden. Ein blinde-Maus-Modell in unsicherer Umgebung wird simuliert, um den speziellen Zustand der Maus zu analysieren. Ein spezieller Platz wird im Labyrinth hinzugefügt, um die Wirkung der ungleich verteilten Abweichungen zu simulieren. Ein spezieller Platz nahe der Katze kann der blinden Maus helfen, um der Katze auszuweichen. Das Verhältnis zwischen den beiden Agenten und der Zustand der Umgebung sind zwei wichtige Faktoren im dualen MDP-Modell.

Abstract

In this work, a dual MDP model is implemented to analyze the relationship of the linked MDP algorithm. Shannon's mouse Theseus is used as a simulation model. A cat is added as the second agent in the maze. The mouse and the cat can influence each other. A new designed labyrinth is used so that the walls of the labyrinth occupy no empty spaces. Three used MDP Algorithms in this paper are value-iteration, Q-learning and TD-learning. The mouse-cat-play is a dynamic decision problem. The goal of the mouse is to find a shortest and safest way to the cheese. The MDP algorithms of the two agents run in parallel.

First, the dual MDP model in the safe environment is simulated. The differences of the reward and the found strategy of the agents are displayed with pictures. The changing of the four rewards can influence the found strategies of the two agents. The dual MDP model can also work under uncertainty. The deviation of the system has more influence for the mouse than the cat. The blind-mouse model is simulated to analyze the action of the mouse, when the mouse is blind. A special place near the cat is added in the maze to help the blind mouse to avoid the danger from the cat. When the deviation of the special place increases, the mouse chooses the safe weg more than before.

Inhaltsverzeichnis

Selbstständigkeitserklärung III
Danksagung IV
Kurzfassung V
Abstract VI
Abbildungsverzeichnis IX
Tabellenverzeichnis X
Diagrammverzeichnis X
Formelgrößen XI
1. Einführung und Motivation 1
2. Grundlagen 5
2.1. Problemdarstellung 5
2.2. Markow-Entscheidungsprozess (MDP) 7
2.3. Grundlagen von Algorithmen 11
2.3.1. Wert-Iteration 11
2.3.2. Q-Lernen 13
2.3.3. Temporal-Difference-Learning 15
2.4. Multiagenten-Systeme 17
2.4.1. Faktoren der Multiagenten-Systeme 17
2.4.2. Modell des Maus-Katze-Spiels 19
3. Duales MDP-Modell in sicherer Umgebung 21
3.1. Spielregeln und Systementwurf 21
3.2. Modell mit gleichen dualen MDP 23
3.2.1. Modell mit dualen Wert-Iterationen 23
3.2.2. Modell mit dualem Q-Lernen 25
3.3. Modell mit gemischten MDP Verfahren 27
3.4. Direkte Belohnungen 31
4. Duales MDP-Modell in unsicherer Umgebung 36
4.1. Modell mit dualem TD-Learning 36
4.2. Modell der blinden Maus 39
4.2.1. Neue Spielregeln und Systemaufbau 39
4.2.2. Modell mit der blinden Maus 41
4.2.3. Modell mit der Katze und der blinden Maus 43

4.3. Modell der blinden Maus mit einem speziellen Platz 46
4.3.1. Spezieller Platz im Labyrinth 46
4.3.2. Modell der blinden Maus mit einem speziellen Platz 47
4.3.3. Modell der dualen MDP Verfahren mit einem speziellen Platz 49
5. Zusammenfassung und Fazit 53
Literaturverzeichnis 56
Abkürzung 58

Abbildungsverzeichnis

Abbildung 1: Systementwurf des kognitiven Systems [Mat17] 2
Abbildung 2: Theseus Maze [Cal50] 5
Abbildung 3: Vergleich der simulierten Labyrinthe....6
Abbildung 4: Platzraum und Zustandsraum 8
Abbildung 5: Vier Möglichkeit der Aktionen 8
Abbildung 6: Beziehungen der vier direkten Belohnungen....10
Abbildung 7: Rechnung des Value-Werts V....12
Abbildung 8: Algorithmus für die Value-Iteration [Wol09] 13
Abbildung 9: Struktur der Q-Matrix....14
Abbildung 10: Rechnungsprozess des Q-Lernens 15
Abbildung 11: Modell des Maus-Katze-Spieles 20
Abbildung 12: Startzustand des Spiels im 6*6 Labyrinth....21
Abbildung 13: Systementwurf des dualen MDP-Modells....22
Abbildung 14: Belohnungsdifferenz des dualen Wert-Iteration Modells 24
Abbildung 15: Strategie des dualen Wert-Iteration Modells 24
Abbildung 16: Belohnungsdifferenzen des dualen Q-Lernens 26
Abbildung 17: Strategie des dualen Q-Lernens....26
Abbildung 18: Belohnungsdifferenzen der gemischten dualen MDP 27
Abbildung 19: Strategie des gemischten dualen MDP Modells 28
Abbildung 20: Belohnungsdifferenzen des gemischten dualen MDP-Modells 29
Abbildung 21: Strategie des gemischten dualen MDP-Modells....29
Abbildung 22: Abweichungsbeeinflussung im dualen TD-Learning Modell....36
Abbildung 23: Belohnungsdifferenz des dualen TD-Learning 37
Abbildung 24: Strategie des dualen TD-Learning 38
Abbildung 25: Vergleich der Belohnungsdifferenz mit drei Alpha-Funktionen 39
Abbildung 26: Struktur des blinden Maus Modells 40
Abbildung 27: Mögliche Routen der Maus zum Käse 41
Abbildung 28: Startzustand des blinden Maus Modells....43

Abbildung 29: Spezieller Platz im Labyrinth 46
Abbildung 30: Startzustand des Modells mit einem speziellen Platz 49

Tabellenverzeichnis

Tabelle 1: R-Matrix für die Maus 10
Tabelle 2: Vergleich der gemischten dualen MDP Modelle 30

Diagrammverzeichnis

Diagramm 1: Beeinflussung der negativen Belohnung der Maus.....31
Diagramm 2: Beeinflussung der positiven Belohnung der Maus 32
Diagramm 3: Beeinflussung der positiven Belohnung der Katze 33
Diagramm 4: Beeinflussung der Belohnung der Käse für die Maus.....34
Diagramm 5: Bewegungsrichtungen des blinden Maus Modells 41
Diagramm 6: Durchschnittlichen Anzahl der Schritte des blinden Maus Modells.....42
Diagramm 7: Gewinn-Wahrscheinlichkeit des blinden Maus Modells.....44
Diagramm 8: Bewegungsrichtungen des blinden Maus Modells 45
Diagramm 9: Durchschnittliche Anzahl der Schritte des blinden Maus Modells 45
Diagramm 10: Veränderung der speziellen Abweichung 48
Diagramm 11: Wahrscheinlichkeit des Gewinnens mit einem speziellen Platz 50
Diagramm 12: Richtungen des Gewinnens der Maus mit einem speziellen Platz.....51
Diagramm 13: Anfangsrichtungen der Maus mit einem speziellen Platz 51

Formelgrößen

z_{tm} oder z_{tk}	Zustand der Maus oder Katze zum Zeitpunkt t
a oder A	Aktion der Agenten oder Aktionsraum
P und S	Platzraum und Zustandsraum
$\delta(z, a)$	Übergangsfunktion
r_{m-}	Negative Belohnung der Maus
r_{m+}	Positive Belohnung der Maus
r_{k-}	Positive Belohnung der Katze
r_{kk}	Positive Belohnung des Käses für die Katze
$z' \, und \, a'$	Nachfolgender Zustand und nachfolgende Aktion
π	Lösungsstrategie
π^*	Optimale Lösungsstrategie
V(z)	Wert / Abgeschwächte Belohnung
γ	Diskontierungsfaktor
α	Gewichtungsfaktor des TD-Learning

1. Einführung und Motivation

‚KI' ist die Abkürzung für künstliche Intelligenz. Die „künstliche Intelligenz (KI)", „kognitives System" und „lernende Maschinen" sind drei populäre Begriffe in der heutigen Zeit. Sie spielen eine wichtige Rolle in der künftigen Entwicklung der Technologie. Die Nachfrage des Marktes nach kognitiven Systemen oder kognitiven Lösungen wächst sehr schnell im 21. Jahrhundert. Sie sind auch wichtige Komponenten im Zukunftsprojekt „Industrie 4.0" in Deutschland.

Was ist ein kognitives System? Eine eindeutige Definition der kognitiven Systeme existiert nicht. Kognition ist die von einem verhaltenssteuernden System ausgeführte Umgestaltung von Informationen [Tob15]. Man modelliert das kognitive System, um menschliches Denken zu simulieren und die Verhaltensweisen zu steuern. Durch den Einsatz der am besten geeigneten Algorithmen ermitteln die kognitiven Systeme die Strategien zur Problemlösung ohne die Lösung selbst zu kennen.

Wie läuft der kognitive Prozess während der Arbeit des kognitiven Systems ab? Zuerst soll das System die Informationen aus der Umwelt wahrnehmen und sammeln. Durch diese Informationen kann das System aus der Umwelt lernen. Der Lernprozess kann mehrmals hintereinander ablaufen. Nach dem Lernen kann das System die beste Strategie finden und eine Handlung ausführen. Unter Umständen kann diese Handlung die Umwelt beeinflussen und verändern.

Durch den kognitiven Prozess kann ein einfaches kognitives System implementiert werden. In Abbildung 1 ist ein Systementwurf eines kognitiven Systems dargestellt [Mat17]. Auf der linken Seite befindet sich der Perzeptionsprozess. Durch die Wahrnehmung kann das System die Information der Umgebung erhalten. Dann wird diese Information analysiert. Anschließend kann das System die Information verstehen und interpretieren. Durch die erhaltende Information und die Algorithmen kann das kognitive System das Verhalten steuern. Danach findet der Aktionsprozess statt. Durch

die Rechnung mit dem Algorithmus kann das System eine Aktion wählen und durchführen. Diese Aktion kann die Umgebung beeinflussen. Dadurch entsteht ein Zyklusverlauf.

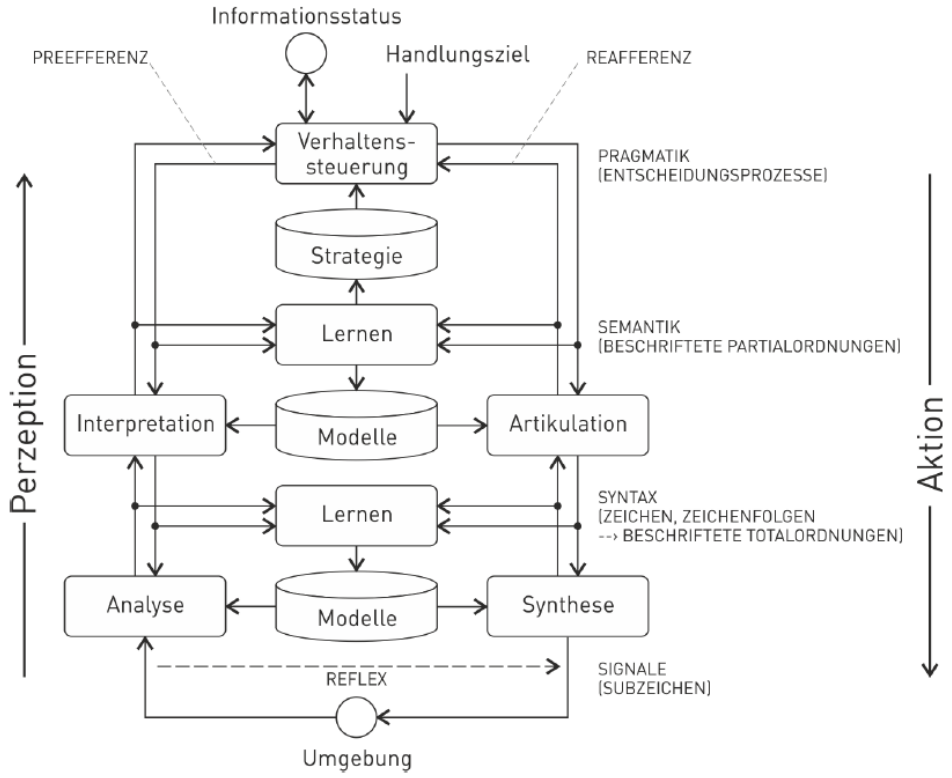

Abbildung 1: Systementwurf des kognitiven Systems [Mat17]

Die Verhaltenssteuerung ist eine wichtige und komplexe Komponente im kognitiven System. Zuerst erhält die Komponente der Verhaltenssteuerung die Information, die aus dem Lernen der Umgebung und der Analyse des Handlungsziels stammt. Einige Algorithmen werden in diesem Teil verwendet. Sie helfen dem Modell eine zielgerichtete Aktion oder Strategie zu finden. Die gefundene Strategie wird im nächsten Schritt durchgeführt. Der Markow Entscheidungsprozess (MEP, im engl.: Markov Decision Process - MDP) ist ein berühmtes mathematisches Modell, welches oft in der Komponente der Verhaltenssteuerung verwendet wird.

Der Markow Entscheidungsprozess wurde nach dem russischen Mathematiker Andrei Andrejewitsch Markow benannt. Dieses Modell beschäftigt sich mit Entscheidungsproblemen. Der MEP ist ein stochastischer Prozess mit diskreten

Zuständen. Jede Entscheidung wird mit zustands- und entscheidungsabhängigen Kosten bewertet. Das Ziel des Modells ist die beste Strategie zu finden, während die Kosten minimiert werden.

Es gibt bereits mehrere Forschungen zu einem Modell mit einem Agenten in einer Labyrinth-Umgebung. Sie bilden eine Grundlage in diesem Forschungsgebiet. In der Realität gibt es manchmal Systeme, die eine Sequenz von Entscheidungsprozessen benötigen oder in denen mehrere Agenten miteinander interagieren. Zwei Agenten haben verschiedene Möglichkeiten miteinander zu interagieren und sich gegenseitig zu beeinflussen. Die Analyse von Modellen mit zwei Agenten ist ein wichtiger Schritt für die zukünftige Forschung.

Diese Masterarbeit baut auf die Masterarbeit „Modellierung und Implementierung einer kognitiven Verhaltenssteuerung" auf [Mey15]. In dieser Arbeit wurde ein Modell von einem Agenten mit dem Markow-Entscheidungsprozess in einer Labyrinth-Umgebung implementiert. Drei grundlegende Verfahren des Lernens wurden dabei dargestellt. Diese drei Verfahren sind Wert-Iteration, Q-Lernen und TD-Learning. Die Verfahren Wert-Iteration und Q-Lernen können in einer sicheren Umgebung angewendet werden und TD-Learning kann in einer unsicheren Umgebung angewendet werden.

Die Hauptaufgabe dieser Masterarbeit besteht darin, eine Analyse und Simulation der Modelle mit dualen MDPs Verfahren durchzuführen. Es wurden viele verschiedene Modelle mit Anwendung von zwei MDP-Agenten implementiert, um das Verhältnis zwischen beiden MDP-Agenten zu analysieren. Das System mit dualen MDP Verfahren wird neu modelliert und ein neuer Zustandsraum wird dargestellt. Die Agenten müssen auch unter einer unsicheren Umgebung arbeiten und Entscheidungen treffen. Deshalb wird das Systemverhältnis unter Unsicherheit beobachtet.

Diese Arbeit besteht aus fünf Kapiteln. Auf die Einführung und Darlegung der Motivation folgt in Kapitel 2 die Darstellung der grundlegenden Kenntnisse und die Theorie der drei MDP-Verfahren. Im dritten Kapitel werden die dualen MDP Modelle in

einer sicheren Umgebung simuliert und die Resultate der Simulationen grafisch dargestellt. Im vierten Kapitel wird die Erweiterung des Modells, durch eine blinde Maus und einen speziellen Platz diskutiert. Hierbei handelt es sich um eine Beobachtung des kognitiven Systems in einer unsicheren Umgebung. Die Arbeit schließt mit einer Zusammenfassung in Kapitel 5.

2. Grundlagen

2.1. Problemdarstellung

Claude Elwood Shannon ist ein amerikanischer Mathematiker und Elektrotechniker. Er ist der Begründer der Informationstheorie. In den fünfziger Jahren entwickelte er die elektrische Maus „Theseus" mit künstlicher Intelligenz. Diese Maus wird von einer lernenden Maschine kontrolliert. Sie befindet sich in einer Labyrinth-Umgebung und ihr Ziel ist es, den Käse im Labyrinth zu finden. Am Anfang befindet sich die Maus am Startplatz. Die Maus lernt das Labyrinth zuerst kennen. Durch die Analyse der wahrgenommenen Informationen kann die Maus schließlich den kürzesten Weg zum Käse finden.

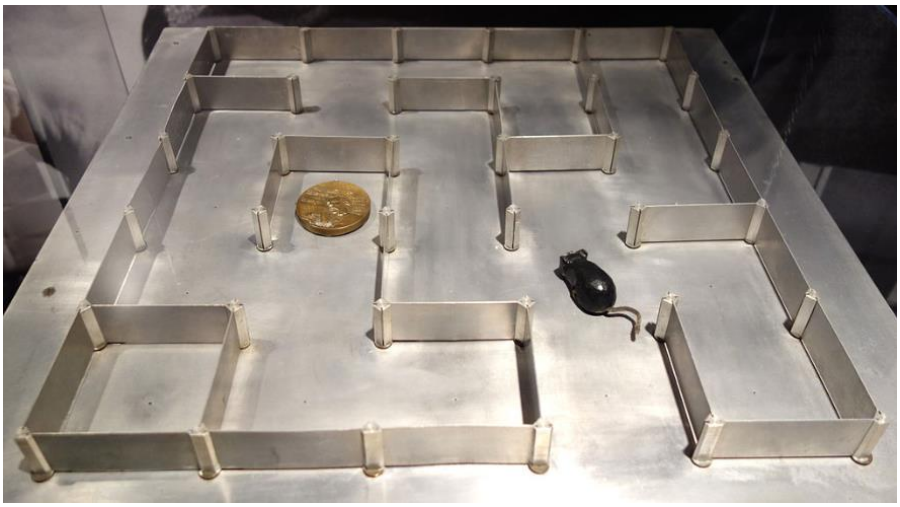

Abbildung 2: Theseus Maze [Cal50]

Dieses Experiment leistet wichtige Beiträge zur Erforschung lernender Maschinen. In diesem Modell wird die Maus als ein Agent betrachtet. Der Käse ist das Zielobjekt der Maus. Das Ziel der Maus ist es, den kürzesten Weg zum Käse zu finden. Das Spiel kann durch viele zusätzliche Bedingungen in vielen verschiedenen Forschungsbereichen erweitert werden.

In dieser Arbeit wird eine Katze als zweiter Agent im Labyrinth hinzugefügt, um das Modell der dualen MDP Verfahren zu analysieren. Das Ziel der Katze ist, die Maus zu

fangen, bevor die Maus den Käse findet. Durch Hinzufügung der Katze, ändert sich das Ziel der Maus. Sie muss die Katze meiden, während sie zum Käse geht. Das Verhältnis zwischen der Maus und der Katze kann die gefundene Strategie beeinflussen. Das Maus-Katze-Spiel wird in einem kognitiven System moduliert.

Das Labyrinth ist eine wichtige Komponente in diesem Modell. Es wird als die Umgebung des kognitiven Systems betrachtet. Das Labyrinth mit dem Käse ist während des ganzen Experiments unveränderlich. Es enthält zwei Bestandteile der Informationen: die frei zugänglichen Plätze und die Position der Wände.

Es gibt viele Methoden in Matlab, um ein Labyrinth zu simulieren. Die am meisten genutzte Methode ist die Speicherung eines Platzes im Labyrinth als eine Zahl in einer zweidimensionalen Matrix. Der Vorteil dieser Methode besteht darin, dass die nachfolgenden Rechnungen einfach sind und die Laufzeit kurz bleibt. Das simulierte Labyrinth bei dieser Methode ist jedoch nicht realitätsnah, da jede Wand im simulierten Labyrinth ein Feld besetzt.

Um das Problem zu lösen, wird eine neue Simulationsmethode des Labyrinths in dieser Arbeit dargestellt. In Abbildung 3 wird ein Vergleich der simulierten Labyrinthe gezeigt. Das linke Labyrinth ist durch die alte Methode simuliert und das rechte Labyrinth ist neu aufgebaut.

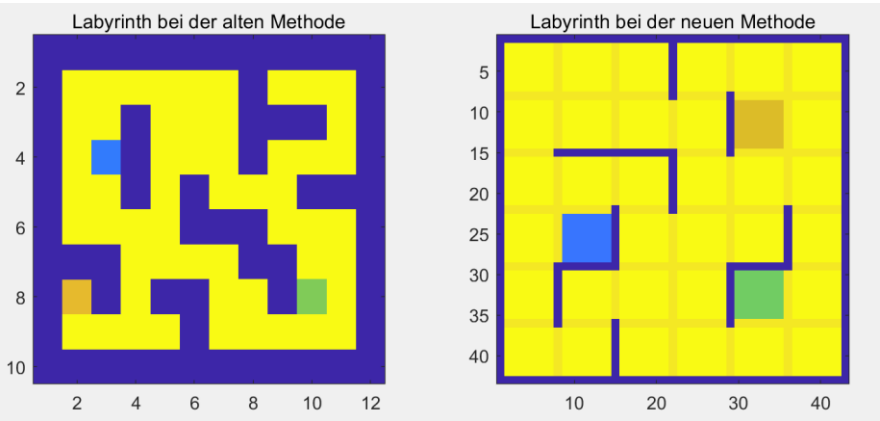

Abbildung 3: Vergleich der simulierten Labyrinthe

Die Informationen des Labyrinths sind nicht mehr in einer zweidimensionalen Matrix

gespeichert, sondern in drei zweidimensionalen Matrizen. Die drei Matrizen speichern drei Teile der Informationen des Labyrinths: die Informationen der frei zugänglichen Plätze des Labyrinths, die Informationen der Wände in den horizontalen Richtungen und die Informationen der Wände in den vertikalen Richtungen. Bei dieser Methode besetzen die Wände des Labyrinths keine freien Felder. Diese verbesserte Methode führt jedoch zu komplexeren Rechnungen als früher.

2.2. Markow-Entscheidungsprozess (MDP)

Der Markow Entscheidungsprozess (MEP, im engl.: Markov Decision Process - MDP) ist ein mathematisches Modell. Der MEP kann wie folgt beschrieben werden [Nic11]:

Ein MEP ist ein Tupel bestehend aus (S, A, P, R, s_0),

- S, Zustandsraum, eine Menge von Zuständen
- A, Aktionsraum, eine Menge von Aktionen
- P, Transitionswahrscheinlichkeit
- R, Belohnungen
- s_0, Startverteilung

Die Maus und die Katze befinden sich zusammen im Labyrinth. Sie beiden sind die Agenten des kognitiven Systems. Die Standorte der Maus und der Katze haben eine relative Beziehung. Der Zustand der Maus z_{tm} und der Zustand der Katze z_{tk} bilden zusammen mit dem Labyrinth einen Zustand des Systems z_t zum Zeitpunkt t. Die Beziehung ist in der Gleichung (2.2.1) dargestellt.

$$z_t = (z_{tm}, z_{tk}) \tag{2.2.1}$$

Nach jeder Aktion a der Maus oder der Katze verändert sich der Zustand des Systems deutlich. Die neue Beziehung ist in der Gleichung (2.2.2) dargestellt.

$$z_{t,a} = \left(z_{tm,a}, z_{tk}\right) oder \left(z_{tm}, z_{tk,a}\right) \tag{2.2.2}$$

Für das Modell mit einem Agenten entspricht die Anzahl der Zustände der Anzahl der frei zugänglichen Felder. Dies gilt jedoch nicht für das Modell mit zwei Agenten. Alle

frei zugängliche Felder des Labyrinths bilden den Platzraum *P* und alle Zustände des Systems bilden den Zustandsraum *S*.

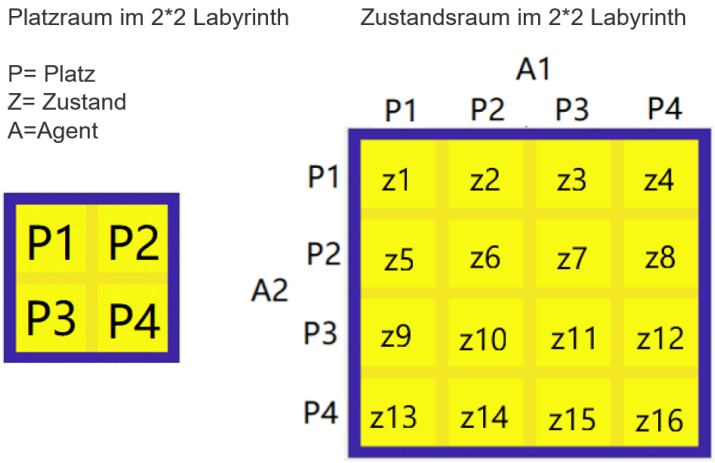

Abbildung 4: Platzraum und Zustandsraum

Die Abbildung 4 zeigt die Beziehung zwischen dem Platzraum P und dem Zustandsraum S in einem 2*2 Labyrinth. Durch diese Abbildung wird deutlich, dass die Anzahl der Plätze nur 4 ist, die Anzahl der Zustände jedoch insgesamt 16. Die Beziehung zwischen der Anzahl der Zustände *z* und der Anzahl der Plätze *p* wird in Gleichung (2.2.3) dargestellt. Hier ist *n* die Anzahl der Agenten im kognitiven System.

$$z = p^n \tag{2.2.3}$$

Je mehr Agenten es gibt, desto komplexer und größer wird der Zustandsraum.

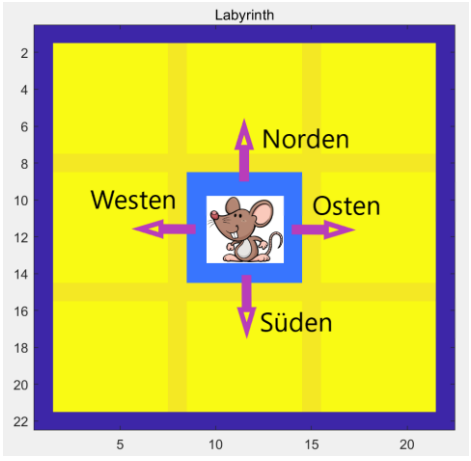

Abbildung 5: Vier Möglichkeiten der Aktionen

Wenn der Agent (Maus oder Katze) sich im Labyrinth befindet, kann er in vier

Richtungen gehen. Eine Aktion *a* besteht also aus vier Möglichkeiten: Aktionsraum *A* = (Westen, Osten, Norden, Süden).

Wenn die Aktion der Maus a_{mt} durchgeführt wird, wird der aktuelle Zustand des Systems z_t zum neuen Zustand z_{t+1} verändert. Durch die Übergangsfunktion $\delta(z_t, a_{mt})$ kann das System den nachfolgenden Zustand z_{t+1} bestimmen.

$$\delta(z_t, a_{mt}) = z_{t+1} \tag{2.2.4}$$

Wenn der Agent von einem Zustand z_t zu einem neuen Zustand z_{t+1} wechselt, erhält er eine direkte Belohnung *r*. Die direkten Belohnungen des Modells mit einem Agenten haben nur zwei Möglichkeiten: null Belohnung (keine Belohnung) und positive Belohnung r_+, wenn die Maus den Käse findet.

Nachdem die Katze dem Modell hinzugefügt wurde, kam eine neue Art der Belohnung hinzu: die negative Belohnung r_- $(r_- < 0)$. Wenn nach der aktuellen Aktion a_{mt} die Maus von der Katze gefangen wurde, erhält die Maus eine negative Belohnung r_{m-} und verliert die Maus somit. Das bedeutet, dass die direkte Belohnung r im Modell mit zwei Agenten eine Beziehung zur Aktion *A* und zum Zustand *Z* hat. Es muss beachtet werden, dass im Modell mit zwei Agenten die direkte Belohnung r nicht direkt mit dem Platz *P*, sondern mit dem Zustand *Z* verbunden ist.

Zusammenfassend kann gesagt werden, dass es vier Arten der direkten Belohnung für das Modell des Maus-Katze-Spiels gibt:

- r_{m+} ist die positive direkte Belohnung für die Maus, welche eintritt, wenn die Maus nach einer Aktion den Käse findet und somit die Maus gewinnt.
- r_{m-} ist die negative direkte Belohnung für die Maus, welche eintritt, wenn die Maus nach einer Aktion a von der Katze gefangen wurde und somit die Maus verliert.
- r_{k+} ist die positive direkte Belohnung für die Katze, welche eintritt, wenn die Katze nach einer Aktion die Maus gefangen hat und somit die Katze gewinnt.

- 0 steht für keine Belohnung und bedeutet, dass nach einer Aktion die Simulation nicht beendet ist.

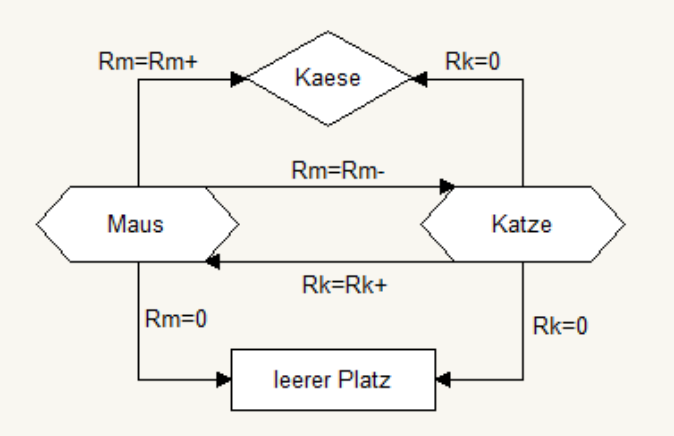

Abbildung 6: Beziehungen der vier direkten Belohnungen

Es gibt zwei wichtige Matrizen im MDP-Modell. Eine ist die Belohnungsmatrix R und die andere ist die Übergangsmatrix P. Die R-Matrix speichert die Werte der direkten Belohnungen jedes Zustands nach jeder Richtung. Die R-Matrizen von Maus und Katze sind unterschiedlich. Sie sind jedoch während des ganzen Experiments nicht veränderlich. Die R-Matrix ist mit den Werten der direkten Belohnungen r verbunden. Wenn die direkten Belohnungen verändert werden, verändert sich auch die R-Matrix jedes Agenten. Tabelle 1 zeigt das Beispiel einer R-Matrix der Maus.

Zustand z_m	Aktion a			
	links	rechts	oben	unten
1	0	0	0	0
2	0	r_{m+}	0	0
3	0	0	r_{m-}	0
.....	.....	.....	.....	.....

Tabelle 1: R-Matrix der Maus

Die P-Matrix speichert die Informationen des nächsten Zustands nach jeder Bewegung. z_t ist der aktuelle Zustand und z_{t+1} ist der nachfolgende Zustand. a_t ist die aktuelle Aktion. Diese Beziehung wird in der Gleichung (2.2.5) gezeigt.

$$P(z_t, z_{t+1}, a_t) = 1 \tag{2.2.5}$$

Aus diesem Grund ist die P-Matrix eine dreidimensionale Matrix. Für die sichere

Umgebung hat jeder Zustand nur einen nachfolgenden Zustand und der Wert in der P-Matrix ist immer 1. Aber für die unsichere Umgebung ist es möglich, dass ein Zustand mehr als einen nachfolgenden Zustand besitzt. Das bedeutet für die unsicher Umgebung, dass jede Zeile der P-Matrix mehr als einen Wert hat，der nicht null ist. Die Summe der Werte in jeder Zeile ist 1.

2.3. Grundlagen von Algorithmen

2.3.1. Wert-Iteration

Das MDP ist ein Modell, das dazu dient Markov Entscheidungsprobleme zu lösen. Der Agent wählt eine Aktion aus dem Aktionsraum aus und bewegt sich von einem Zustand zu einem anderen Zustand. Zum Schluss erreicht der Agent den Zielplatz und die Kosten des zurückgelegten Wegs sollen dabei minimiert werden. Die Wert-Iteration ist ein Algorithmus, der zur Lösung des MDP-Modells eingesetzt wird. Durch den Algorithmus Wert-Iteration werden die Werte aller Zustände bewertet.

Ein zeitdiskreter stochastischer Prozess mit abzählbarem Zustandsraum heißt Markov-Kette, wenn alle Zustände die Markov-Eigenschaft haben. Die Markov-Eigenschaft besagt, dass die zukünftige Entwicklung des Prozesses nur von dem zuletzt beobachteten Zustand abhängt und von der Vergangenheit unabhängig ist [Wal13]. Wegen dieser Eigenschaft kann der Value-Wert des aktuellen Zustands $V(z_t)$ durch die Gleichung (2.3.1) berechnet werden.

$$V(z_t) = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \cdots = \sum_{i=0}^{\infty} \gamma^i \cdot r_{t+i} \tag{2.3.1}$$

V ist hier die abgeschwächte Belohnung und γ ist der abgeschwächte Parameter des Modells ($0 < \gamma \leq 1$). r_t ist die direkte Belohnung zum Zeitpunkt t.

Die folgende Gleichung (2.3.2) heißt Bellmann-Gleichung. Durch sie können die Value-Werte aller Zustände berechnet werden.

$$\mathrm{V}(z) = \max_a \big[r(z,a) + \gamma \cdot V\big(\delta(z,a)\big)\big] \tag{2.3.2}$$

a ist hier die Aktion des Agenten und $a \in A$. Die direkte Belohnung r ist abhängig vom aktuellen Zustand z und der Aktion a. Die Belohnungen aller Zustände sind in der R-Matrix gespeichert. Während der Rechnungen kann der Wert der Belohnung aus der R-Matrix erhalten werden. $V\big(\delta(z,a)\big)$ ist der Value-Wert des nachfolgenden Zustands.

Die Rechnung kann bei jedem Zustand des Modells starten. Danach laufen die Rechnungen aller Zustände in der Reihenfolge. So entsteht ein zyklischer Prozess. Die Value-Werte aller Zustände bilden die V-Matrix. Durch mehrmaligen Ablauf der Bewertungsprozesse kann die V-Matrix zum Schluss konvergiert werden. Die optimale Strategie π^* kann durch die folgende Gleichung (2.3.3) bestimmt werden.

$$\pi^*(z) = \arg\max_a \big[r(z,a) + \gamma \cdot V^*\big(\delta(z,a)\big)\big] \qquad (2.3.3)$$

Abbildung 7 zeigt ein einfaches Beispiel der Rechnung in einem 3*3 Labyrinth. Die Maus befindet sich im Zustand 5 (z_5). Der Value-Wert des Zustands 5 ist V.

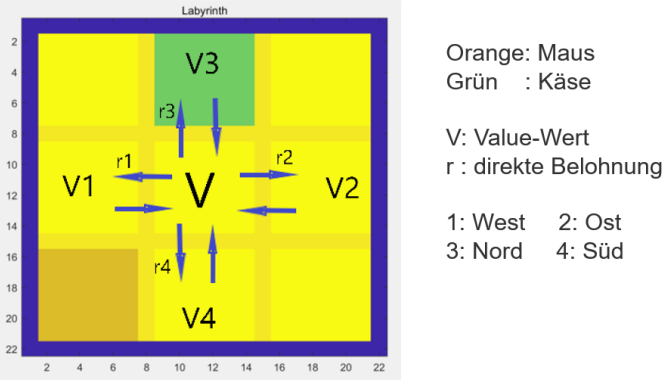

Abbildung 7: Rechnung des Value-Werts V

Welche Richtung ist in diesem Beispiel die optimale Bewegungsrichtung für die Maus? Zuerst müssen die direkten Belohnungen $r(z_5, a_{i=1\cdots4})$ in den vier Richtungen aus der R-Matrix gesucht werden (in Abbildung 7 ist $r_i = r(z_5, a_i)$). Der abgeschwächte Parameter γ ist konstant (z.B. $\gamma = 0.9$). Danach werden die Value-Werte der vier folgenden Zustände durch die V-Matrix bestimmt (V1 bis V4). Die Bewertungswerte für die vier Richtungen können jetzt berechnet werden.

$$V_{z_5,a_1} = r_1 + 0.9 * V_1 \quad \text{und} \quad V_{z_5,a_2} = r_2 + 0.9 * V_2$$

$$V_{z_5,a_3} = r_3 + 0.9 * V_3 \quad \text{und} \quad V_{z_5,a_4} = r_4 + 0.9 * V_4$$

Anschließend wird die Bewegungsrichtung a_{i^*} mit dem maximalen Wert von V_{z_5,a_1} bis V_{z_5,a_4} gesucht. Der berechnete Value-Wert $V_{z_5,a_{i^*}}$ in der Richtung a_{i^*} ist der neue Value-Wert des Zustands 5. Der nachfolgende Zustand ist durch die Übergangsfunktion $\delta(z_5,\ a_{i^*})$ bestimmt. Eine neue Rechnung wird gestartet.

Dieses Verfahren wird Wert-Iteration genannt. Der Ablauf des Algorithmus ist in Abbildung 8 angezeigt.

Wert-Iteration()

Für alle z ϵ Z

$V(z) = 0$

Wiederhole

Für alle z ϵ Z

$$V(z) = \max_a [r(z,a) + \gamma \cdot V(\delta(z,a))]$$

Abbruch V(z) stagniert

Abbildung 8: Algorithmus für die Value-Iteration [Wol09]

Das Verfahren der Wert-Iteration hat zwei Vorteile. Der erste Vorteil besteht darin, dass der Agent mit hundertprozentiger Sicherheit eine Strategie finden kann. Der zweite Vorteil ist, dass die Rechnungszeit keine lange Zeit in Anspruch nimmt. Das Verfahren hat jedoch auch einen Nachteil: Die gefundene Strategie ist nicht immer optimal. Durch den Algorithmus der Wert-Iteration können die Aktionen der folgenden Zustände nicht bewertet werden. Das Verfahren des Q-Lernens stellt somit ein verbessertes Verfahren im Vergleich zur Wert-Iteration dar.

2.3.2. Q-Lernen

Die Wert-Iteration kann durch die Bewertung aller Zustände eine Strategie finden. Hierbei ist die gefundene Strategie jedoch nicht immer optimal. Wenn es mehr als eine optimale Bewegungsrichtung für einen Zustand gibt, bedeutet das, dass die Bewertungswerte der nachfolgenden Zustände gleich sind. Es kann nicht entschieden

werden, welche Richtung besser ist. Das Q-Lernen ist ein verbesserter Algorithmus und kann das oben genannte Problem lösen.

Das Q-Lernen (auch „Q-Learning" engl.) bewertet die vier Bewegungsrichtungen des aktuellen Zustands. Es gibt einen Bewertungswert für jede Bewegungsrichtung an. Das heißt, dass es vier Bewertungswerte für jeden Zustand gibt. Diese Bewertungswerte bilden die Q-Matrix. Die Abbildung 9 zeigt den Prozess bei der Bildung der Q-Matrix. 0 bedeutet hier, dass es eine Wand in der Bewegungsrichtung gibt und der Agent nicht in diese Richtung gehen kann.

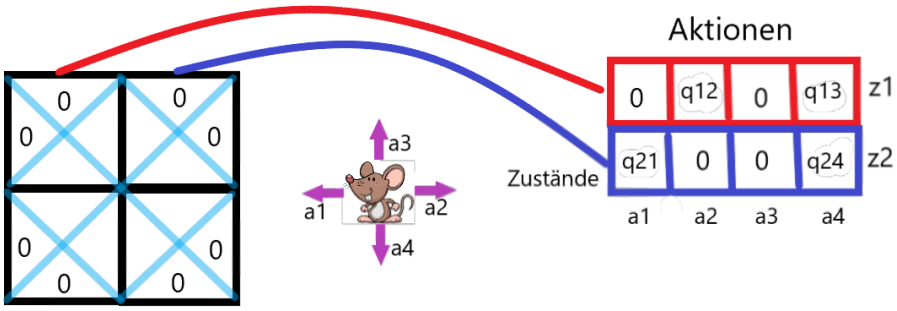

Abbildung 9: Struktur der Q-Matrix

Die folgende Gleichung (2.3.4) heißt „action value function". Wenn der aktuelle Zustand z_t und die wählbaren Aktionen A gegeben sind, kann die Q-Wert für eine bestimmte Aktion a_t berechnet werden. $\delta(z,a)$ ist hier die Übergangsfunktion und a' ist die optimale Aktion des folgenden Zustands. Das bedeutet, dass für den nachfolgenden Zustand $\delta(z,a) = z'$ der Q-Wert in der Richtung a' maximal ist.

$$Q(z_t, a_t) = r(z_t, a_t) + \gamma \cdot \max_{a'} Q(\delta(z,a), a') \qquad (2.3.4)$$

Die folgende Abbildung 10 zeigt den Rechnungsprozess des Q-Lernens. Zuerst soll die Q-Matrix initialisiert werden. Alle Werte der Q-Matrix können die Werte 0 oder 1 annehmen. Aus dem Vergleich aus der vorhergehenden Arbeit [Mey15] geht hervor, dass 1 besser als 0 ist, weil die mit 1 initialisierte Q-Matrix schneller konvergiert werden kann.

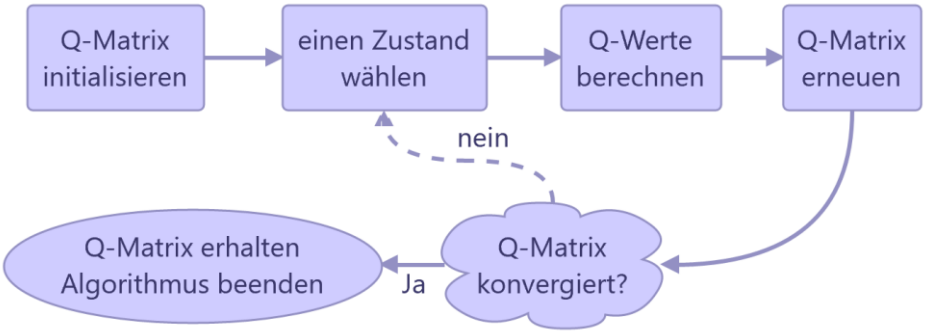

Abbildung 10: Rechnungsprozess des Q-Lernens

Am Anfang wird ein zufälliger Zustand aus dem Zustandsraum gewählt. Die Q-Werte des gewählten Zustands in vier Richtungen werden berechnet. Nach der Rechnung wird die Q-Matrix erneuert. Der folgende gewählte Zustand kann zufällig sein. Er kann auch durch das vorherige Lernen gewählt werden. Der Prozess kann durch eine Funktion kontrolliert werden. In dieser Arbeit ist die Episode nur auf 1 gesetzt. Der Zyklus endet, wenn die Q-Matrix konvergiert wird.

Nach dem Ablauf des Algorithmus erhält das System eine Q-Matrix, die alle Zustände in allen Bewegungsrichtungen bewertet. Die optimale Strategie kann durch die folgende Gleichung (2.3.5) bestimmt werden. π^* steht für die optimale Strategie. Für den Zustand z wird die Aktion a mit dem maximalen Q-Wert $Q(z,a)$ gewählt.

$$\pi^* = \underset{a}{\operatorname{argmax}}\, Q(z,a) \tag{2.3.5}$$

Die vier Q-Werte eines Zustands sind nicht gleich. Deshalb kann durch das Q-Lernen eine optimale Strategie gefunden werden, vorausgesetzt es gibt nur einen Agenten. Manchmal benötigt das Q-Lernen eine längere Laufzeit als die Wert-Iteration, um die Q-Matrix zu konvergieren.

2.3.3. Temporal-Difference-Learning

Die beiden Verfahren Wert-Iteration und Q-Lernen können nur in sicherer Umgebung angewendet werden. In der Realität gibt es jedoch immer Abweichungen des Systems. Wenn die Maus beispielsweise nach Norden gehen möchte, geht sie aufgrund von

Abweichungen des Systems nach Westen oder Osten. Der Algorithmus Temporal-Difference-Learning kann das Markow-Entscheidungsproblem in der unsicheren Umgebung lösen.

TD-Learning ist die Abkürzung für Temporal-Difference-Learning und stellt eine Erweiterung des Q-Lernens dar. In diesem Algorithmus wird die Übergangsmatrix (P-Matrix) geändert, weil es in der unsicheren Umgebung möglich ist, dass jeder Zustand mehrere nachfolgenden Zustände hat.

Wegen der Abweichung des Systems kann die Q-Matrix durch die Anwendung der Gleichung (2.3.4) nicht konvergiert werden. Deshalb muss die Gleichung in zwei Teile wie folgende Gleichung (2.3.6) geteilt werden. α ist hier der Gewichtungsfaktor und abhängig vom Iterationsschritt. In der Gleichung (2.3.6) ist der erste Teil die vorherige Belohnung und der zweite Teil die nachfolgende Belohnung.

$$Q_n(z,a) = (1-\alpha)Q_{n-1}(z,a) + \alpha_n \cdot \left[r(z,a) + \gamma \cdot \max_{a'} Q(\delta(z,a),a')\right] \tag{2.3.6}$$

Der Gewichtungsfaktor α ist variabel und liegt zwischen 0 und 1. Er ist abnehmend bei den einnehmenden Iterationsschritten. Die Gleichung (2.3.7) stammt aus der MDP-Toolbox in Matlab und n ist hier die Seriennummer der Iterationsschritte. In dieser Gleichung ist n nicht abhängig vom Zustand z oder der Aktion a. Die Gleichung (2.3.8) stammt aus der Arbeit von Herrn Meyer [Mey15]. $b_n(z,a)$ ist die Seriennummer der Iterationsschritte für den Zustand z und die Aktion a. Der Gewichtungsfaktor α ist abhängig von dem folgenden Zustand und die ausgeführte Aktion, um eine gleichwertige Behandlung aller Übergangsfunktionen zu gewährleisten.

$$\alpha_n = \frac{1}{\sqrt{n+2}} \tag{2.3.7}$$

$$\alpha_n = \frac{1}{1+b_n(z,a)} \tag{2.3.8}$$

Die folgende Gleichung ist die Umformung der Gleichung (2.3.6). Nach der Umformung ist die Gleichung (2.3.9) besser für die Modulation geeignet, weil der erste Teil keine Beziehung mehr zum Gewichtungsfaktor α hat.

$$Q_n(z,a) = Q_{n-1}(z,a) + \alpha_n \cdot \left[r(z,a) + \gamma \cdot \max_{a'} Q(\delta(z,a), a') - Q_{n-1}(z,a) \right] \qquad (2.3.9)$$

Wenn der Gewichtungsfaktor α=1 ist, entspricht die Gleichung (2.3.9) der Gleichung (2.3.4).

2.4. Multiagenten-Systeme

2.4.1. Faktoren der Multiagenten-Systeme

Die Abkürzung MMDP steht für Multi-Agenten MDP und wird auch als Multiagenten-System bezeichnet. Im Multiagenten-System gibt es mehre Agenten, die parallel in sequentiellen Entscheidungsprozessen agieren [Joh05]. Wie ist ein Multi-Agenten System aufgebaut? Zunächst besteht das System aus zwei oder mehreren Agenten. Die Agenten befinden sich zusammen in der Umgebung. Sie haben gleiche oder verschiedene Aktionsräume. Ihre Aktionen können einander beeinflussen. Sie haben entweder ein gleiches Ziel oder verschiedene Ziele. Die Beziehungen der Multiagenten können die Eigenschaften des Systems verändern.

In der heutigen Zeit spielen Multiagenten-Systeme eine immer wichtigere Rolle. Viele Wissenschaftler arbeiten in diesem Forschungsbereich. Craig Boutilier implementierte ein koordiniertes Multiagenten-System [Cra96]. Im Jahre 2001 diskutierte Snehasis das Multiagenten-System mit eingeschränkter Agenten-Kommunikation. Das Problem wurde bei der Anwendung des Coping Algorithmus gelöst [Sne01]. Bei Jean-Francois Raskin und Ocan Sankur wurde ein MMDP-Modell in einer mehrfachen Umgebung analysiert [Jea14]. Joris Scharpff schlug ein TI-MMDP (transition-independent Markov Decision Processes) Modell vor. Sie entwickelte eine neue Methode, das CRG (conditional return graph) heißt, wodurch das Modell optimiert und das Problem gelöst wurde [Jor15]. In der Arbeit von Shayegan Omidshafiei wurden Multi-Task Multi-Agenten Probleme analysiert und gelöst [Sha17].

Wie können die Multiagenten-Systeme beschrieben werden? Die Frage lässt sich nur

schwer beantworten, da bisher keine eindeutige Definition gefunden wurde. Zusammenfasst lässt sich sagen, dass es vier wichtige Aspekte für das Multiagenten-System gibt:

Der erste Aspekt betrifft die Agenten und ihre Beziehungen. Das MMDP System besitzt mehre Agenten, welche verschiedene Beziehungen untereinander haben. Sie können koordinativ, gegensätzlich oder unabhängig voneinander sein. Sie können die anderen Agenten beobachten oder nicht. Verschiedene Verhältnisse der Agenten bilden verschiedene Systeme.

Die direkten Belohnungen eines Agenten repräsentiert das Verhältnis zu anderem Agenten. Eine Belohnung zeigt das Gewicht eines beobachteten Faktors für diesen Agenten. Deshalb sind die Belohnungen eines Systems sehr wichtig für die Modellierung.

Der zweite Aspekt ist die Umgebung des Systems. Die Umgebung kann statistisch oder veränderlich bei verschiedenen Bedingungen sein. Manchmal ist die Umgebung statistisch während des Lernens. Der Lernprozess muss sich nicht nach jeder Bewegung wiederholen. Das Umgebungsbereich eines jeden Agenten kann ganz gleich, ganz unabhängig oder teilweise überlappend sein.

Der dritte Aspekt ist der Algorithmus des Lernens. Wert-Iteration, Q-Lernen und TD-Learning sind drei umfangreiche angewendete Algorithmen. Sie können die meisten normalen Probleme lösen. Für einige spezielle Probleme des Multiagenten-Systems gibt es noch viele erweiterte Algorithmen aus dem Q-Lernen oder anderen Algorithmen.

Der letzte Aspekt betrifft die Aktionen der Agenten. Jeder Agent hat seinen eigenen Aktionsraum. Eine Aktionsraum kann auch die anderen Aktionsräume überlappen. Die Aktionen werden in der Komponente Verhaltenssteuerung kontrolliert. Nach dem Lernen kennt der Agent die Umgebung. Danach wählt der Agent eine Reihe von Aktionen aus. Die gewählte Aktion ist nicht nur abhängig von der Umgebung, sondern auch abhängig vom aktuellen Systemzustand, das bei der letzten Aktion des anderen

Agenten verändert wurde. Das ist ein großer Unterschied zum Single-Agent-System.

2.4.2. Modell des Maus-Katze-Spiels

Das Spiel, das in dieser Arbeit simuliert wird, ist ein Maus-Katze-Spiel. Das Spiel hat zwei Agenten, eine Maus und eine Katze. Sie befinden sich zusammen in einer Labyrinth-Umgebung. Außerdem gibt es auch einen Käse mit festem Standort. Die Katze möchte die Maus so bald wie möglich fangen. Die Maus möchte den Käse auf dem sichersten und kürzesten Weg erreichen und muss dabei der Katze aus dem Weg gehen. Die Maus und die Katze können das ganze Labyrinth erkunden und kennenlernen. Sie können auch einander beobachten. Die Katze gewinnt das Spiel, wenn sie die Maus fängt. Die Maus gewinnt das Spiel, wenn sie den Käse findet, bevor sie von der Katze gefangen wird. Es muss beachtet werden, dass die Länge des gefundenen Weges nicht am wichtigsten für die Maus ist. Der wichtigste Faktor für die Maus ist ihre Sicherheit. Wie kann das Spiel moduliert werden?

Zunächst muss erwähnt werden, dass die zwei Agenten das Labyrinth kennenlernen können und das Labyrinth statisch und nicht veränderlich ist. Es sind zwei Agenten vorhanden, welche gegensätzlich zu einander sind. Deshalb sind die Lernprozesse der beiden Agenten unabhängig voneinander. Die Maus und die Katze können gleiche oder verschiedene MDP Algorithmen eingesetzt werden. Das Labyrinth und die Standorte von Maus und Katze bilden einen Zustandsraum. Nach dem Lernen bei MDP Verfahren kann eine gemeinsame Strategie gefunden werden und die Simulation endet, wenn die Maus oder die Katze gewinnt. Der Systementwurf des Maus-Katze-Spieles wird in folgender Abbildung 11 gezeigt.

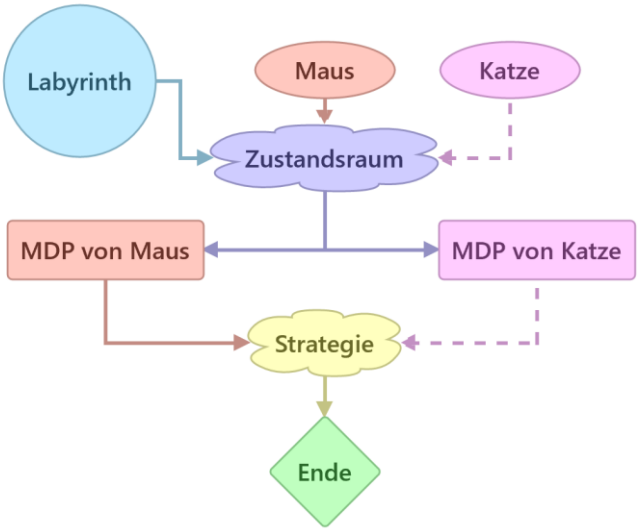

Abbildung 11: Modell des Maus-Katze-Spieles

3. Duales MDP-Modell in sicherer Umgebung

3.1. Spielregeln und Systementwurf

Um das duale MDP System zu analysieren, soll zuerst ein Systemmodell aufgebaut werden. Die elektrische Maus bei Shannon wurde gewählt, um das System zu simulieren. Früher gab es nur eine Maus im Spiel und diese Maus war der erste Agent. Jetzt soll das Speil erweitert werden, indem der zweite Agent hinzugefügt wird. Die Spielregeln müssen aus diesem Grund verändert werden.

Die Maus und die Katze können sich im Labyrinth frei bewegen. Die Katze möchte die Maus so bald wie möglich fangen, bevor die Maus den Käse findet. Die Maus möchte den Käse finden, aber sie muss auf ihre Sicherheit achten und die Katze umgehen. Zu Beginn stehen Maus und Katze an verschiedenen Plätzen im Labyrinth. Die Abbildung 12 zeigt den Startzustand der Simulation.

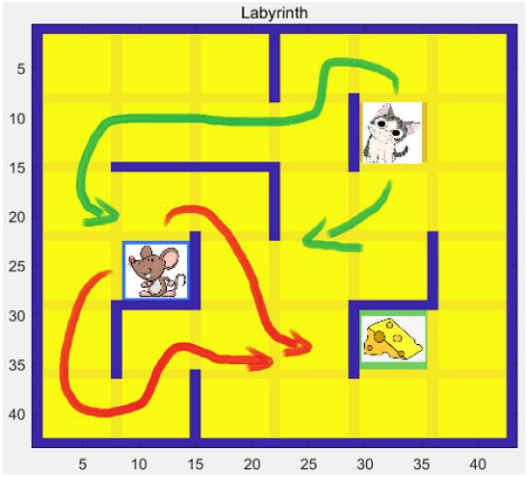

Abbildung 12: Startzustand des Spiels im 6*6 Labyrinth

In der Simulation des dualen MDP-Modells in der sicheren Umgebung wird ein 6*6 Labyrinth gewählt. In diesem Labyrinth hat die Maus mehr als zwei Wege zum Käse. Der kurze Weg geht nahe am Startplatz der Katze vorbei und ist somit gefährlicher für die Maus ist. Der lange Weg hat mehr Schritte, ist aber sicherer. Welchen Weg wählt die Maus? Das ist ein schwieriges Entscheidungsproblem.

Die neuen Spielregeln lauten wie folgt:

a) Die zwei Agenten können in vier Richtungen gehen. Die vier Richtungen sind Westen, Osten, Norden und Süden.

b) Am Anfang des Spiels geht die Maus zuerst. In jedem Schritt kann die Maus sich nur um einen Platz im Labyrinth bewegen.

c) Die Katze geht nach dem Schritt der Maus und sie kann sich auch nur um einen Platz bewegen.

d) Wenn die Katze die Maus fängt, endet die Simulation sofort. Die Katze gewinnt und die Maus verliert.

e) Wenn die Maus den Käse findet, bevor die Katze die Maus findet, endet die Simulation auch sofort. Die Maus gewinnt und die Katze verliert.

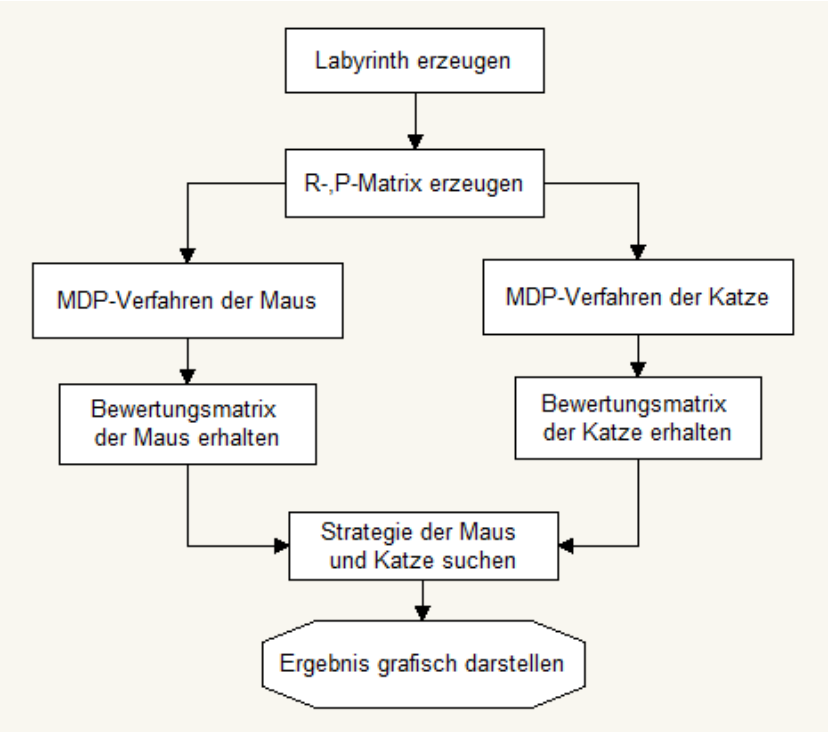

Abbildung 13: Systementwurf des dualen MDP-Modells

Der Systementwurf in der Simulation ist in Abbildung 13 dargestellt. Zuerst wird ein Labyrinth mit Maus, Katze und Käse erzeugt. Das ist der Startzustand des Systems. Dann werden die Belohnungsmatrix (R-Matrix) und die Übergangsmatrix (P-Matrix) durch die Informationen des Labyrinths erzeugt. Die Information der Plätze und Zustände werden auch dokumentiert. Danach laufen die zwei MDP-Verfahren parallel.

Nach den Rechnungen erhält das System die Bewertungsmatrizen (V-Matrix oder Q-Matrix). Durch die Bewertungsmatrizen kann das System eine Strategie finden. Zum Schluss wird das Ergebnis grafisch dargestellt.

Durch die Abbildung 13 wird deutlich, dass die zwei MDP-Verfahren voneinander getrennt sind. Deshalb ist das System ein paralleles System. Die zwei parallelen MDP-Verfahren können gleich oder verschieden sein. Im anschließenden Kapitel 3.2 wird das Modell mit zwei gleichen MDP-Verfahren dargestellt. In Kapitel 3.3 wird dann das Modell mit zwei verschiedenen MDP-Verfahren dargestellt. Zum Schluss werden in Kapitel 3.4 die direkten Belohnungen des Modells analysiert.

3.2. Modell mit gleichen dualen MDP

3.2.1. Modell mit dualen Wert-Iterationen

Die Wert-Iteration ist ein MDP-Verfahren und sie ist die Grundlage für das Q-Lernen und TD-Learning Verfahren. Zuerst wird ein duales MDP-Modell mit dualen Wert-Iteration Verfahren aufgebaut. Vor der Simulation werden einige wichtige Parameter aufgestellt.

Der erste Parameter ist die direkte Belohnung r. Die positiven direkten Belohnungen für die Maus und Katze sind auf 1 gesetzt. Die negative Belohnung für die Maus ist auf -10 gesetzt. Der abgeschwächte Parameter γ nimmt einen Wert zwischen 0 und 1 an. Durch die frühere Analyse wurde deutlich, dass je kleiner der abgeschwächte Parameter ist, desto schneller sinkt die Belohnungsdifferenz [Mey15]. In diesem Modell ist er auf 0.9 gesetzt, um die Veränderung der Belohnungsdifferenz klarer zu beobachten. Die Anfangswerte der Value-Matrizen sind alle auf 0 gesetzt. Die maximalen Iterationsschritte der Wert-Iteration sind durch die Funktion in der MDP Toolbox berechnet.

Die folgende Abbildung 14 zeigt den Ablauf der Belohnungsdifferenz des dualen Wert-Iteration Modells. In den meisten Simulationen sind die Anzahl der Iterationen für die

Maus und die Anzahl der Iterationen für die Katze fast gleich. Die Kurve der Belohnungsdifferenz sinkt sehr schnell. Das bedeutet, dass die beiden Value-Matrizen der Maus und der Katze schnell konvergieren können. Zum Schluss kann eine Strategie durch die beiden V-Matrizen gefunden werden.

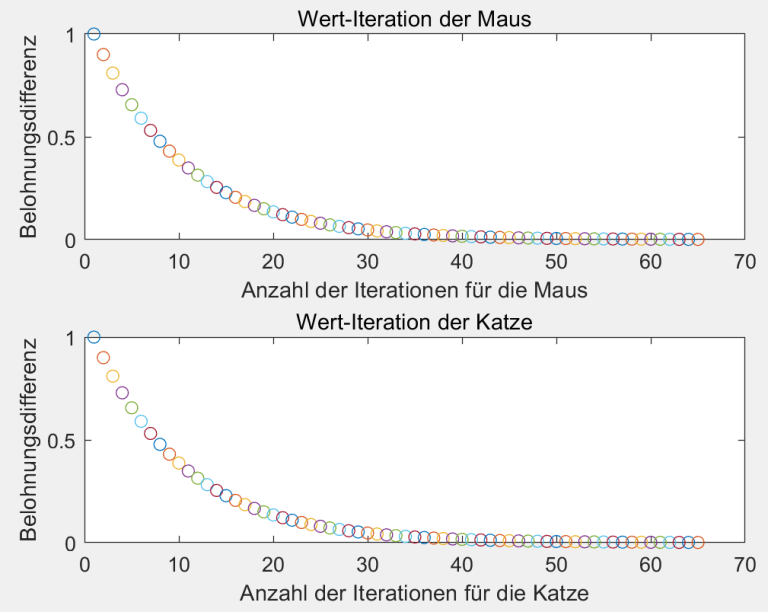

Abbildung 14: Belohnungsdifferenz des dualen Wert-Iteration Modells

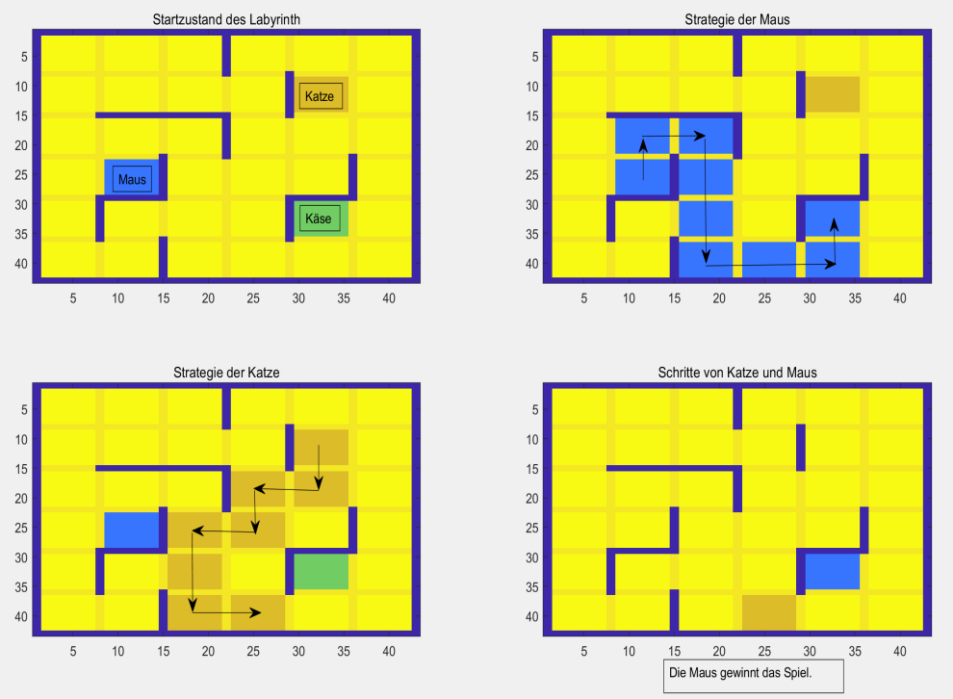

Abbildung 15: Strategie des dualen Wert-Iteration Modells

Die Abbildung 15 zeigt eine gefundene Strategie des Modells mit dualen Wert-

Iterationen. Die erste Abbildung (links oben) zeigt den Startzustand der Simulation. Die zweite Abbildung (rechts oben) zeigt die Route der Maus. Es ist klar, dass die Maus zum Käse geht. Die dritte Abbildung (links unten) zeigt die Route der Katze. Durch diese Abbildungen wird deutlich, dass die Katze ihre Bewegungsrichtung nach dem aktuellen Standort der Maus wählt. Zuerst geht die Katze nach links, weil die Maus links steht. Danach geht die Katze nach rechts, weil die Maus in diesem Moment rechts von der Katze steht. Die letzte Abbildung (rechts unten) zeigt den Bewegungsprozess der Simulation dynamisch. Zum Schluss gewinnt die Maus in diesem Beispiel und die Katze verliert. Wenn die Simulation mehrmals abläuft, kann die Maus sowohl gewinnen als auch verlieren. Die Wahrscheinlichkeit des Gewinnens steht in Beziehung zum Labyrinth und auch zu den Startplätzen der beiden Agenten.

3.2.2. Modell mit dualem Q-Lernen

Das Q-Lernen Verfahren ist ein Algorithmus, der aus der Wert-Iteration verbessert wurde. Das Q-Lernen bewertet mit der Q-Matrix die Bewegungsrichtungen. Die Bewertung der Bewegungsrichtungen ist besser als die Bewertung der nachfolgenden Zustände. Die Wahrscheinlichkeit, eine optimale Strategie durch Q-Lernen zu finden, ist höher als durch die Anwendung der Wert-Iteration.

Dieses Modell hat dieselben direkten Belohnungen wie das Modell der dualen Wert-Iterationen. Der abgeschwächte Parameter γ ist auch auf 0.9 gesetzt. Die Anfangswerte der Value-Matrix sind alle auf 0 gesetzt und die Anfangswerte der Q-Matrix sind alle auf 1 gesetzt. Die Anzahl der Iterationen muss beim Q-Lernen vorgegeben werden. Nach dem Ende der Iteration muss die Q-Matrix konvergieren.

Die folgende Abbildung 16 zeigt die Belohnungsdifferenzen des dualen Q-Lernens. Die Belohnungsdifferenzen der beiden Agenten haben zuerst zugenommen und danach abgenommen. Durch diese Abbildung wird sichtbar, dass die Q-Matrix der Maus schneller als die Q-Matrix der Katze bei der gleichen Anzahl der Iterationen konvergiert, weil die Maus eine negative Belohnung hat. Wenn der absolute Wert der negativen

Belohnung zunimmt, sinkt die Belohnungsdifferenz schneller. Je höher die Anzahl der Iterationen ist, desto besser konvergiert die Q-Matrix. Deshalb kann man das berechnete Ergebnis durch Erhöhen der Anzahl der Iterationen verbessern.

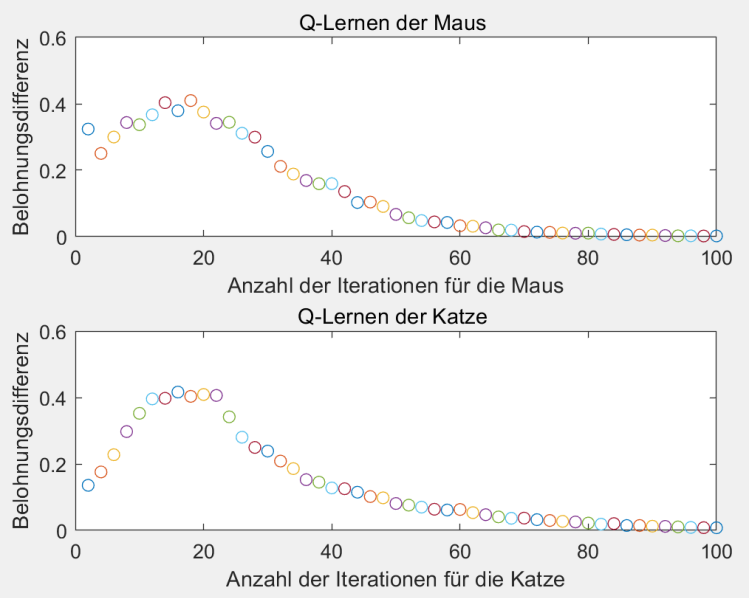

Abbildung 16: Belohnungsdifferenzen des dualen Q-Lernens

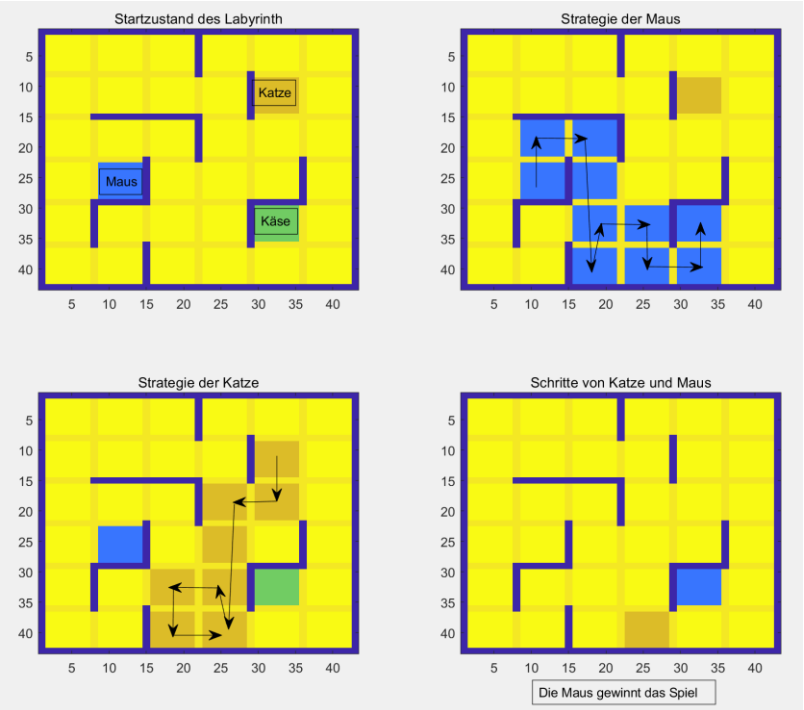

Abbildung 17: Strategie des dualen Q-Lernens

Die Abbildung 17 zeigt eine gefundene Strategie des dualen Q-Lernens. Hier wurde

das gleiches Labyrinth gewählt. Außerdem stehen Maus und Katze an den gleichen Startplätzen. Die gefundenen Strategien weichen jedoch von den vorherigen ab.

Die Schritte der Maus im Modell des dualen Q-Lernens sind länger als im Modell der dualen Wert-Iterationen, weil die Katze einen Weg geht, der für die Maus gefährlich ist. Die Maus passt sich der Situation an und umgeht die Katze. Nach jedem Durchlauf der Simulation des Spiels ist die berechnete Q-Matrix nicht gleich, weil der gewählte Zustand jeder Iteration zufällig ist. Die Anzahl der Iterationen beeinflusst die berechnete Q-Matrix ebenfalls. Ungleiche Q-Matrizen führen dazu, dass die gefundene Strategie jeder Simulation unterschiedlich ist.

3.3. Modell mit gemischten MDP Verfahren

Die dualen MDP Modelle können nicht nur durch zwei gleichen MDP-Verfahren, sondern auch durch zwei verschiedenen MDP-Verfahren aufgebaut werden. Hier werden die Modelle mit gemischten MDP Verfahren getestet, um die Eigenschaften der Systeme zu analysieren.

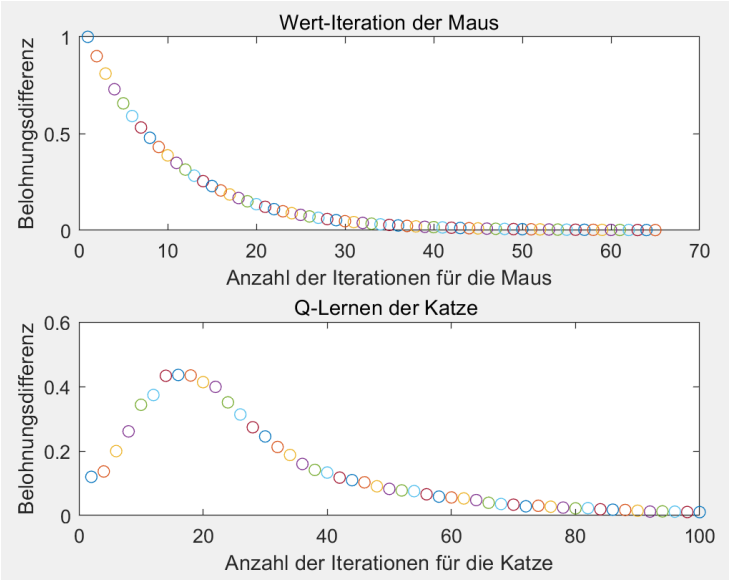

Abbildung 18 Belohnungsdifferenzen der gemischten dualen MDP-Modell

Für die Maus wird der Algorithmus der Wert-Iteration und für die Katze der Algorithmus des Q-Lernens gewählt. Die zwei Lernprozesse laufen parallel. Abbildung 18 zeigt die

Belohnungsdifferenzen des Modells.

Durch diese Abbildung wird erkennbar, dass die beiden Kurven der Belohnungsdifferenzen den im Kapitel 3.2 dargestellten Kurven stark ähneln. Die bedeutet, dass ein Lernprozess nicht vom anderen beeinflusst wird. Man kann einen geeigneten Algorithmus für jeden Agenten wählen, um die besseren Eigenschaften des Systems zu erhalten.

Abbildung 19 zeigt eine gefundene Strategie des gemischten WQ-Modells (Maus: Wert-Iteration, Katze: Q-Lernen). Die Maus und die Katze wählen ihre optimalen Wege. Zum Schluss gewinnt die Maus in diesem Beispiel.

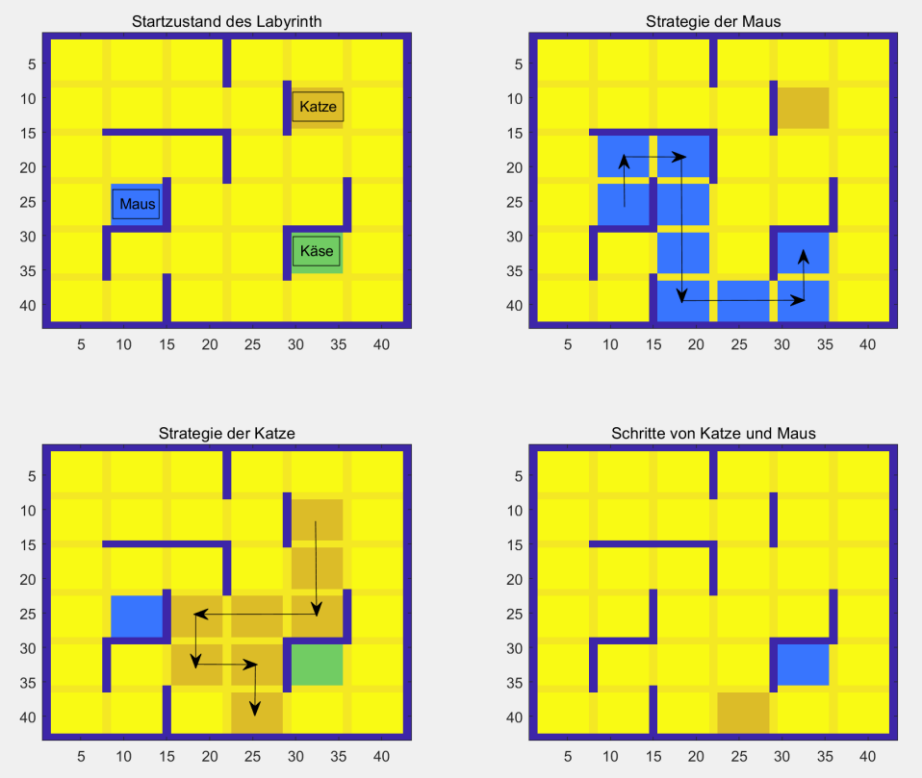

Abbildung 19: Strategie des gemischten dualen MDP-Modells

Um die gemischten dualen MDP Modelle zu vergleichen, wird ein weiteres MDP-Modell betrachtet. Für die Maus werden der Algorithmus des Q-Lernens und für die Katze der Algorithmus der Wert-Iteration gewählt. Die folgende Abbildung 20 zeigt die Belohnungsdifferenz des neu gebauten Modells. Die beiden Kurven entsprechen den Eigenschaften jedes Algorithmus.

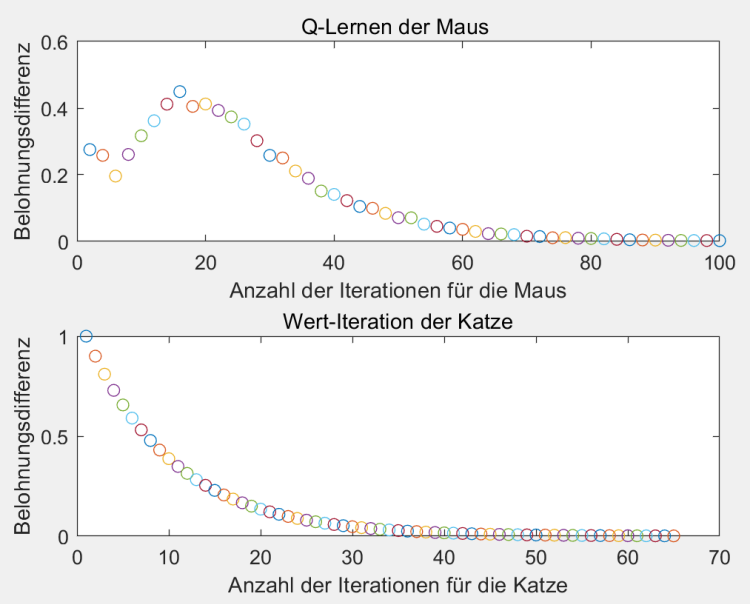

Abbildung 20: Belohnungsdifferenzen des gemischten dualen MDP-Modells

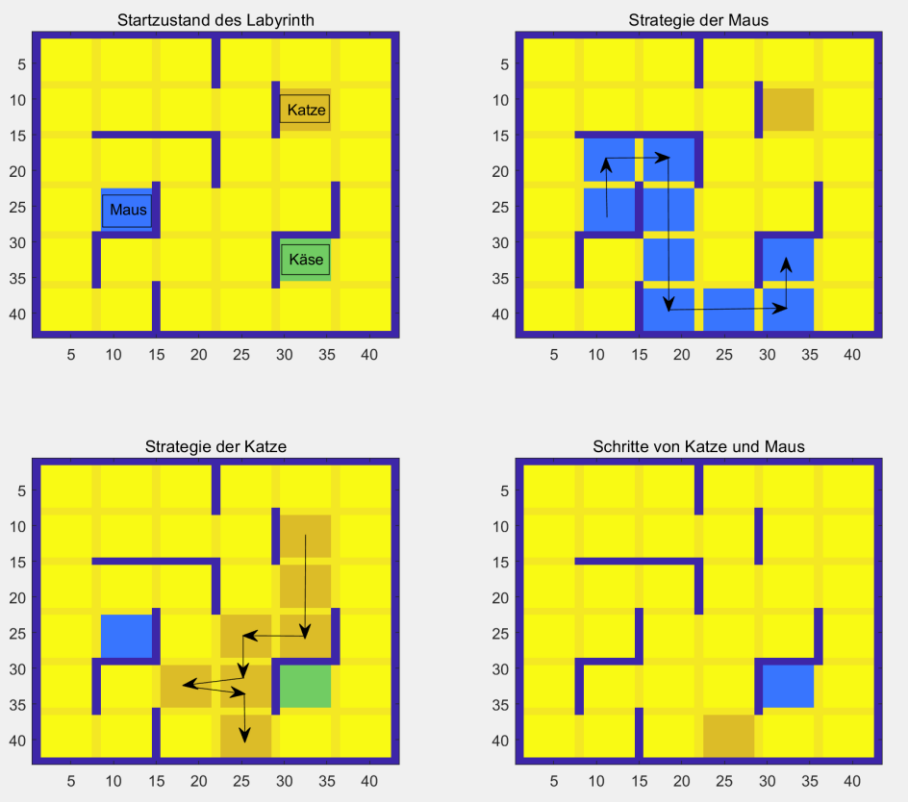

Abbildung 21: Strategie des gemischten dualen MDP-Modells

Die Abbildung 21 zeigt eine gefundene Strategie des Modells. Durch die verschiedenen MDP-Verfahren kann die Maus einen optimalen Weg finden. Zum Schluss gewinnt die Maus in diesem Beispiel.

Die folgende Tabelle 2 zeigt den Vergleich der drei dualen MDP-Modelle. Jedes System läuft hundert Mal ab, um ein statistisches Ergebnis zu berechnen. Durch das statistische Ergebnis werden die Eigenschaften der Modelle besser analysiert. (M=Maus, K=Katze, WI=Wert-Iteration, QL=Q-Lernen)

	M=QL, K=QL	M=QL, K=WI	M=WI, K=QL
Maus gewinnt	91.7%	100%	91.3%
Durchschnittliche Schritte	9.8	10.01	9.75

Tabelle 2: Vergleich der dualen MDP-Modelle

Die erste Zeile der Daten zeigt die Wahrscheinlichkeit des Gewinnens durch die Maus. Wenn für die Katze der Algorithmus Q-Lernens gewählt wird, kann die Maus nur 91% der Simulationen gewinnen. Sie kann jedoch 100% der Simulationen gewinnen, wenn der Algorithmus der Wert-Iteration für die Katze gewählt wird. Daraus kann man schließen, dass das Q-Lernen für die Katze besser als die Wert-Iteration ist, weil das Q-Lernen die Zustände des Systems besser als die Wert-Iteration bewerten kann.

Die zweite Zeile der Daten zeigt die durchschnittliche Anzahl der Schritte der Maus, wenn die Maus gewinnt. Der kürzeste Weg bei diesem Labyrinth beträgt 9 Schritte. Die maximale Anzahl der Schritte wird beim zweiten Modell erzeugt. Durch die Analyse wird deutlich, dass die Maus eine längere Route wählt, um die Gefahr durch die Katze zu vermeiden, sodass sie mehr gewinnt. Der Weg beim dritten Modell ist ein bisschen kürzer als der Weg beim ersten Modell. Die Maus wählt eine kürzere Route, wenn der Algorithmus der Wert-Iteration für die Maus angewendet wird. Das bedeutet, dass das Q-Lernen besser als die Wert-Iteration geeignet ist, um die negative Beeinflussung (die Gefahr durch die Katze) zu vermeiden.

Zusammenfassend kann gesagt werden, dass das in dieser Arbeit aufgebaute duale MDP-System ein paralleles System ist. Die zwei gewählten MDP-Verfahren für die beiden Agenten können in der sicheren Umgebung parallel ablaufen und beeinflussen einander nicht. Deshalb kann man für jeden Agenten einen geeigneten Algorithmus wählen. In dieser Simulation gibt es zwei Aspekte für die Maus, aber nur einen Aspekt

für die Katze. Deshalb beeinflusst der Algorithmus mehr die Maus als die Katze. Man kann die Leistung des Systems durch die Anwendung eines besseren Algorithmus verbessern.

3.4. Direkte Belohnungen

Während der Simulation ergab sich eine interessante Situation. Die gesetzten Belohnungen r des Systems können die Leistung des Modells deutlich beeinflussen. Durch die folgenden vier Diagramme der statistischen Daten kann man diese Beeinflussung besser analysieren. Es gibt vier Belohnungen in diesem Modell:

1) Negative Belohnung der Maus r_{m-}
2) Positive Belohnung der Maus r_{m+}
3) Positive Belohnung der Katze r_{k+}
4) Positive Belohnung des Käses für die Katze r_{kk}

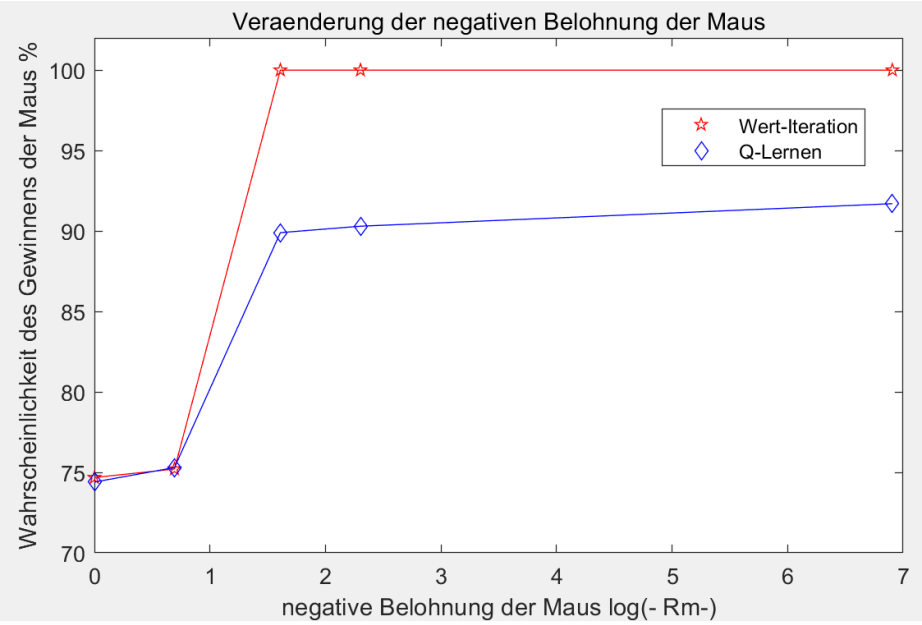

Diagramm 1: Beeinflussung der negativen Belohnung der Maus

Das Diagramm 1 zeigt die Beeinflussung der negativen Belohnung der Maus r_{m-}. Die Belohnungen r_{m+} und r_{k+} sind auf 1 gesetzt und verändern sich nicht. Die Belohnung r_{m-} sinkt von -1 auf -999. Die negative Belohnung der Maus steht für die Angst der Maus vor den gefährlichen Faktoren. Je höher der Absolutwert von r_{m-} ist,

desto mehr Angst hat die Maus vor der Katze.

Wenn r_{m+} und der Absolutwert r_{m-} gleich sind, bedeutet dies, dass der positive Faktor und negative Faktor gleiche Beeinflussung haben. Wenn der Absolutwert r_{m-} größer als r_{m+} ist, bedeutet dies, dass die Maus in erster Linie die Katze vermeiden muss, bevor sie zum Käse geht. Deshalb steigt die Wahrscheinlichkeit des Gewinnens für die Maus deutlich, wenn r_{m-} von -1 auf -2 sinkt. Die Wahrscheinlichkeit des Gewinnens für die Maus steigt langsam, wenn r_{m-} von -2 auf -999 sinkt. Der Aufstieg ist bei einem kleinen Labyrinth sehr klein.

Das Diagramm 2 zeigt die Beeinflussung der positiven Belohnung der Maus r_{m+}. Die Belohnung r_{m-} ist auf -10 gesetzt und r_{k+} ist auf 1 gesetzt. Sie verändern sich in diesen Gruppen Tests nicht. Die Belohnung r_{m+} steigt von 1 auf 999. Die positive Belohnung der Maus steht für den Wunsch der Maus, den Käse zu finden. Je höher der Wert von r_{m+} ist, desto größer ist der Wunsch der Maus, den Käse zu finden.

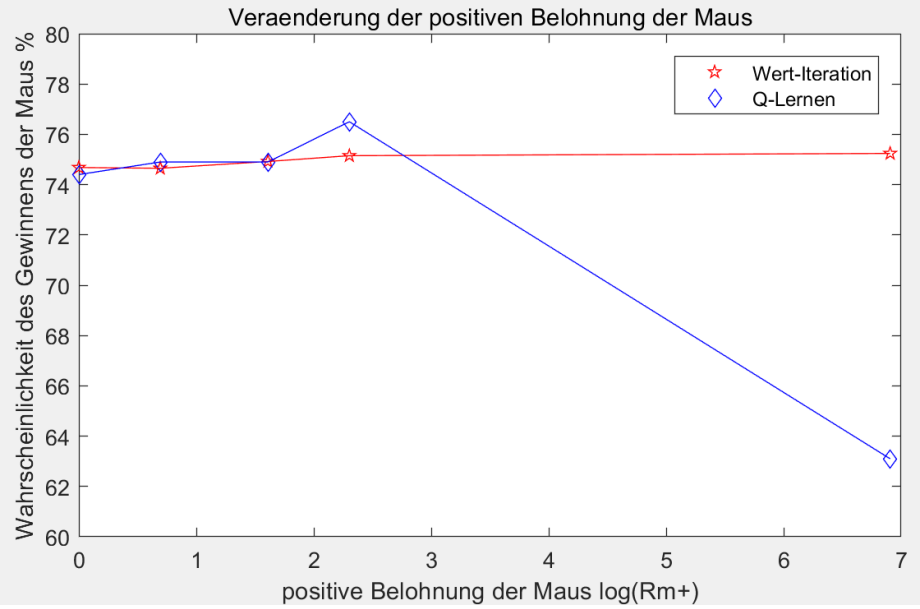

Diagramm 2: Beeinflussung der positiven Belohnung der Maus

Durch das Diagramm kann man finden, dass die Wahrscheinlichkeit des Gewinnens der Maus für die Welt-Iteration ziemlich nicht verändert. Das bedeutet, dass der Einstieg der Belohnung r_{m+} auf die Leistung des Systems der dualen Welt-Iterationen

nicht beeinflusst. Im Modell der dualen Q-Lernens verändert die Kurve zuerst nicht. Wenn r_{m+} sehr groß ist, sinkt die Wahrscheinlichkeit des Gewinnens. Wenn der Wunsch der Maus sehr hoch ist, kann die Maus die Gefahr der Katze nicht rechtzeitig vermeiden.

Das Diagramm 3 zeigt die Beeinflussung der positiven Belohnung der Katze r_{k+}. Die Belohnung r_{m-} ist auf -10 gesetzt und r_{m+} ist auf 1 gesetzt. Sie verändern sich nicht. Die Belohnung r_{k+} steigt von 1 auf 999. Die positive Belohnung der Katze steht für den Wunsch der Katze, die Maus zu fangen. Je höher der Wert von r_{k+} ist, desto größer ist der Wunsch der Katze, die Maus zu fangen.

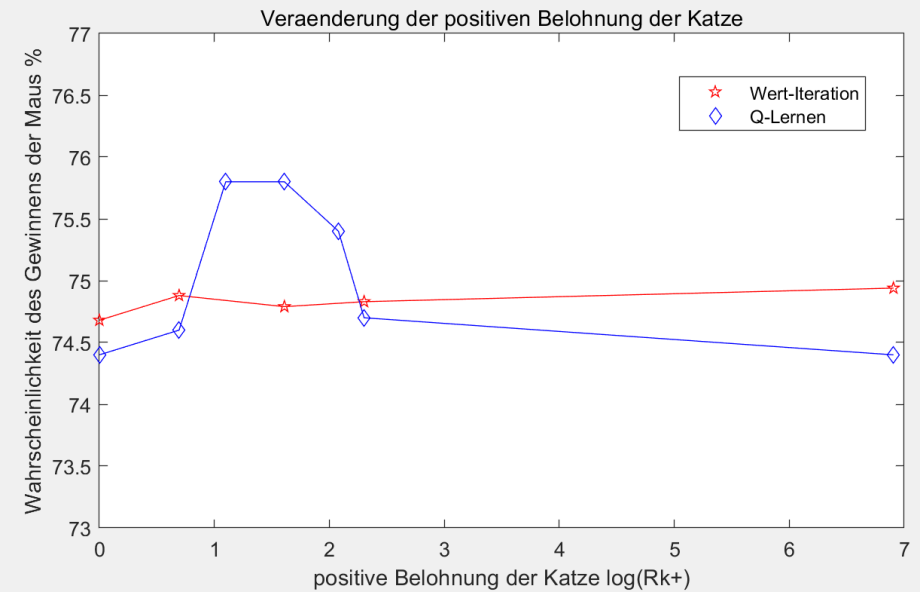

Diagramm 3: Beeinflussung der positiven Belohnung der Katze

Das Diagramm zeigt, dass die Kurven der beiden Systeme horizontal bleiben. Es gibt eine kleine Spitze beim Modell des dualen Q-Lernens. Im Allgemeinen beeinflusst die Veränderung der positiven Belohnung der Katze das System jedoch nicht.

Im Simulationsmodell beobachtet die Katze nur den stehenden Platz der Maus. Sie achtet nicht auf den Platz des Käses. Deshalb gewinnt die Maus in den meisten Fällen. Wenn die Katze auch den Käse beobachtet, bedeutet dies, dass die Katze den Zielplatz der Maus kennt. Die Strategie der Katze muss sich also stark verändern. Hier wird eine

positive Belohnung des Käses für die Katze r_{kk} im System hinzugefügt. Wenn die Katze die positive Belohnung des Käses besitzt, sucht die Katze die Maus nahe dem Käse.

Das Diagramm 4 zeigt die Beeinflussung der Belohnung des Käses für die Katze r_{kk}. Die Belohnungen r_{m+} und r_{k+} sind auf 1 gesetzt und r_{m-} ist auf -10 gesetzt. Sie verändern sich nicht. Die Belohnung r_{kk} steigt von 0 auf 10. Diese Belohnung bedeutet, dass die Katze den Zielplatz der Maus kennt und immer beobachtet. Je höher der Wert von r_{kk} ist, desto größer ist die Beobachtung des Zielplatzes der Maus durch die Katze.

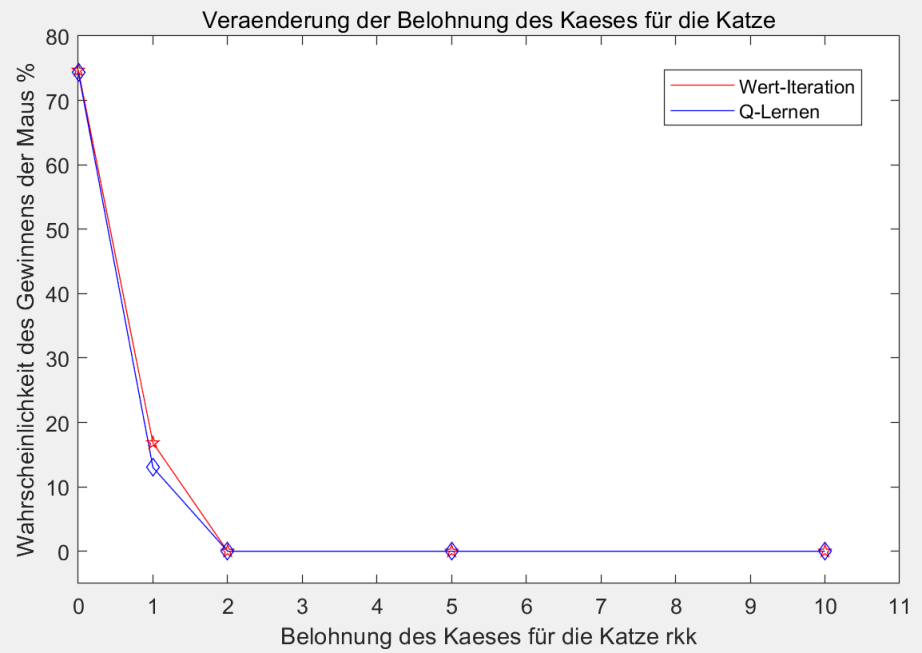

Diagramm 4: Beeinflussung der Belohnung des Käses für die Katze

Das Diagramm 4 zeigt, dass die Kurve sehr schnell beim Einstieg der Belohnung r_{kk} sinkt. Wenn die Belohnung r_{kk} 1 ist, gewinnt die Katze mehr als die Maus. Die Katze hat mehr Chance, die Maus zu fangen, wenn sie den Zielplatz der Maus kennt und sie die Maus nahe beim Zielplatz sucht. Wenn die Belohnung r_{kk} höher als r_{k+} (hier $r_{k+} = 1$) ist, kann die Maus nicht gewinnen, weil die Katze am Zielplatz der Maus steht und wartet. Zum Schluss kann die Maus die Katze nicht vermeiden und sie verliert das Spiel.

In den folgenden diskutierten Modellen sind die Belohnungen des Modells festgesetzt. Die Belohnungen r_{m+} und r_{k+} sind auf 1 gesetzt und r_{m-} ist auf -10 gesetzt. Die Belohnung r_{kk} ist auf 0 gesetzt. Der Einfluss des Käses auf die Katze wird hier ignoriert.

Durch die vorherige Analyse zeigt sich, dass die Belohnungen des Modells die Leistung stark beeinflussen können. Dies gilt besonders für die Multiagenten-Systeme, weil die Multiagenten-Systeme mehr Arten der Belohnungen haben. Wenn ein Multiagenten-System implementiert werden soll, können geeignete Belohnungen des Modells gewählt werden, um die Leistung des Systems zu maximieren.

4. Duales MDP-Modell in unsicherer Umgebung

Im Kapitel 3 wurden duale MDP-Modelle in der sicheren Umgebung diskutiert. Die meisten Systemumgebungen sind jedoch nicht statisch. Es gibt sehr oft Abweichungen, welche die Bewegungen der Maus und der Katze beeinflussen können. Diese Abweichungen in den dualen MDP Modellen können auch das Verhalten bei verknüpften Prozessen beeinflussen. Dieser Einfluss muss beachtet und analysiert werden.

4.1. Modell mit dualem TD-Learning

Der TD-Learning Algorithmus wird im unsicheren dualen MDPs Modell angewendet, um die Systemabweichungen in der unsicheren Umgebung zu modellieren. Zuerst wird das normale System mit dualen TD-Learning Algorithmen dargestellt. Anschließend werden zwei spezielle Modelle im folgenden Kapiteln diskutiert.

Wenn es eine 10% Abweichung im System gibt, werden die Bewegungsrichtungen der Agenten verändert. Beispielweise gilt die Maus in einer sicheren Umgebung zu 100% nach oben. Bei Abweichungen wird sie nur zu 80% nach oben und zu jeweils 10% in zwei andere Richtungen in der Nähe (Westen und Osten im Beispiel) gehen. Die Abbildung 22 zeigt diese Abweichungsbeeinflussung im dualen TD-Learning Modell.

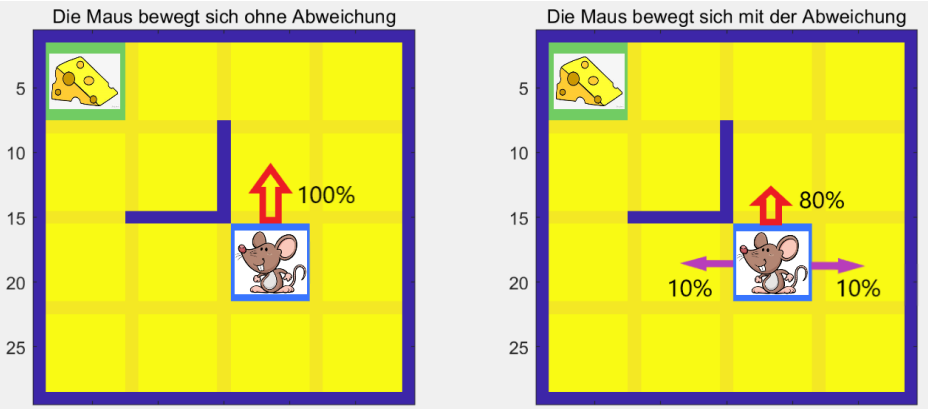

Abbildung 22: Abweichungsbeeinflussung im dualen TD-Learning Modell

Der TD-Learning Algorithmus stellt eine Erweiterung des Q-Lernens dar. Durch die Funktionen des Q-Lernens kann die Bewertungsmatrix in unsicherer Umgebung nicht konvergieren. Deshalb wird eine Gleichung des Parameters Alpha in der TD-Learning angewendet, dadurch kann die Bewertungsmatrix konvergieren. Abbildung 23 zeigt die Belohnungsdifferenzen des TD-Learning Prozesses.

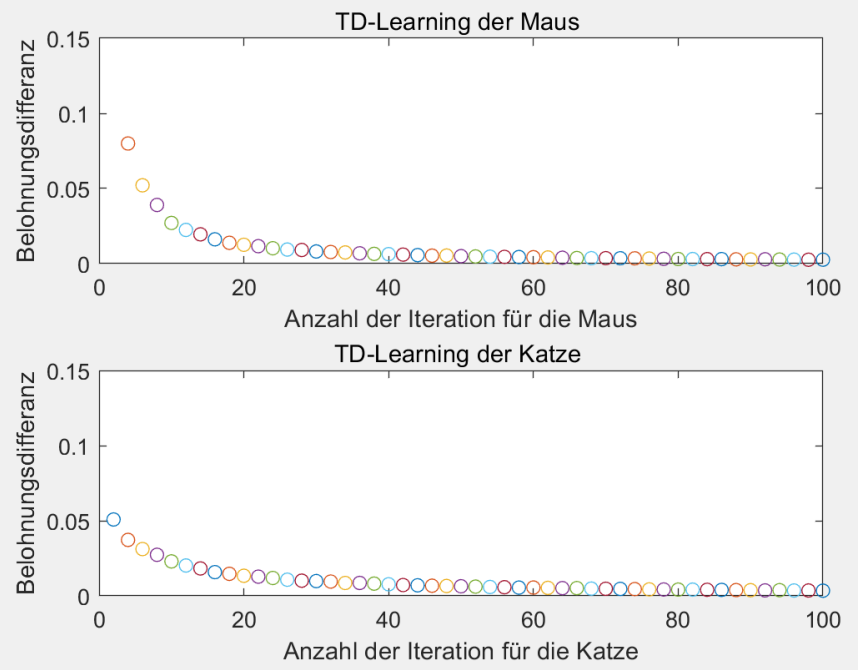

Abbildung 23: Belohnungsdifferenz des dualen TD-Learning

In der Abbildung wird deutlich, dass die Kurven der Belohnungsdifferenzen immer sinken, dies geschieht aber sehr langsam. Wegen der Abweichung sinkt die Belohnungsdifferenz des TD-Learning langsamer als beim Q-Lernen. Beim TD-Learning werden mehr Iterationen benötigt, um die Q-Matrix zu konvergieren. Wenn die Q-Matrix nicht konvergiert ist, kann das Modell unter Umständen keine optimale Strategie finden. Je größer die Abweichung ist, desto mehr Iterationen werden benötigt. Mehr Iterationen bedeuten längere Laufzeit.

Die folgende Abbildung 24 zeigt eine gefundene Strategie des dualen TD-Learning mit einer10% Abweichung des Systems. Um das Verhalten der verknüpften Prozesse besser zu analysieren, wird ein neues Labyrinth in der Simulation erstellt. Im neuen Labyrinth sind die optimalen Wege der Maus in beide Richtungen ziemlich gleich.

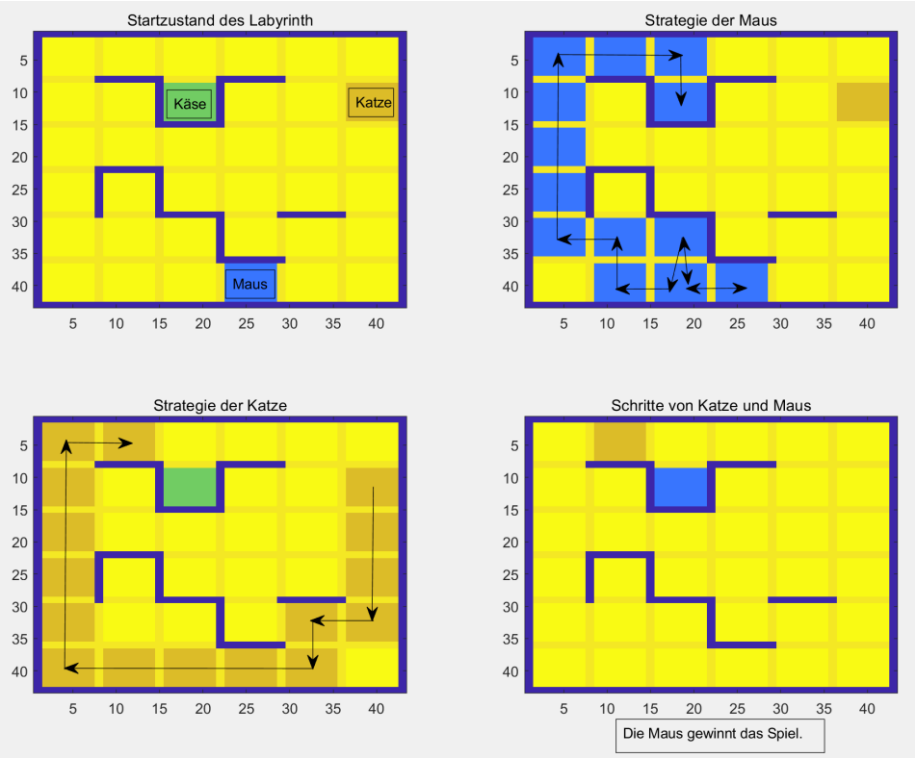

Abbildung 24: Strategie des dualen TD-Learning Modells

In der Abbildung der Strategie der Katze wird deutlich, dass die Katze immer vorwärts und nie rückwärts geht. Da die Katze nur ein Ziel hat, gibt es weniger Einfluss durch die Abweichung. Die Katze und der Standort des Käses, die gegensätzlich sind, können die Bewegungen der Maus beeinflussen. Die Abweichung hat einen größeren Einfluss auf die Maus als auf die Katze. Aus der zweiten Abbildung wird ersichtlich, dass die Maus zwei Mal zwischen zwei Plätzen hin- und zurückzugeht. Diese Situation ist hierbei durch die Abweichung verursacht. Je mehr gegenseitige Faktoren den Agenten beeinflussen, desto größer ist der Einfluss der Abweichung auf das Modell.

Die Alpha-Funktionen können die Leistung des Systems beeinflussen. Die folgende Abbildung 25 zeigt einen Vergleich der drei Alpha-Funktionen. Die erste Funktion ist bei Herrn Meyer gegeben [Wer15]. Die zweite Funktion kommt aus der MDP Toolbox in Matlab. Die dritte Funktion ist 1 und sie wird im Q-Lernen angewendet.

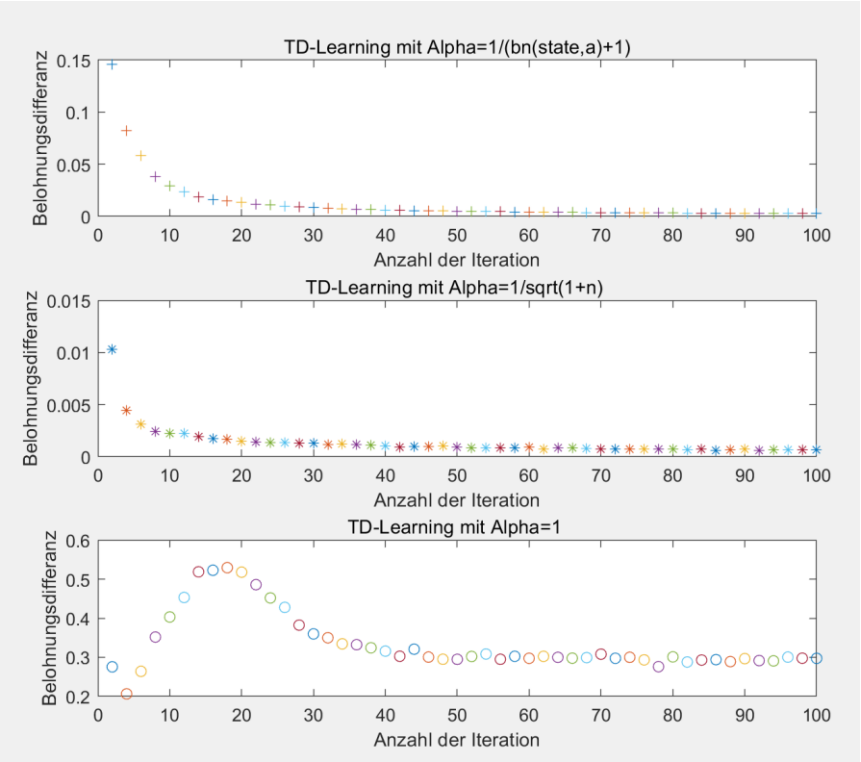

Abbildung 25: Vergleich der Belohnungsdifferenz mit drei Alpha-Funktionen

Durch diese Abbildung wird deutlich, dass die dritte Belohnungsdifferenz bei Alpha=1 immer 0.3 bleibt und die Q-Matrix nicht konvergieren kann. Die ersten und zweiten Kurven der Belohnungsdifferenz gehen auf 0 zu, sodass diese beiden Funktionen genutzt werden können. Die erste Kurve sinkt schneller als die zweite Kurve. Das bedeutet, dass die erste Alpha-Funktion für das Modell besser geeignet ist.

4.2. Modell der blinden Maus

4.2.1. Neue Spielregeln und Systemaufbau

Die Maus in den vorherigen Modellen kennt das Labyrinth und kann die Aktionen der Katze beobachten. In der Realität ist sich der Agent häufig nicht der Gefahr in seiner Umgebung bewusst. In diesem Spiel bedeutet es, dass die Maus blind ist. In diesem Fall kann die Maus das Labyrinth kennen, jedoch nicht die Katze. Durch das neue Modell mit der blinden Maus kann das neue Verhalten der verknüpften Prozesse analysiert werden.

Durch die Blindheit der Maus hat die Katze keinen Einfluss auf die Maus. Der neue Zustandsraum der Maus ist entspricht einem Ein-Agent-MDP-Modell. Dieser Zustandsraum ist kleiner. Deshalb ist der Lernprozess der Maus schneller als der der Katze. So kann die Laufzeit der Simulation gespart werden.

Weil der Zustandsraum des blinden Maus Modells verändert ist, muss die Struktur des Modells auch verändert werden. Die neue Struktur des Systems ist in der folgenden Abbildung 26 dargestellt.

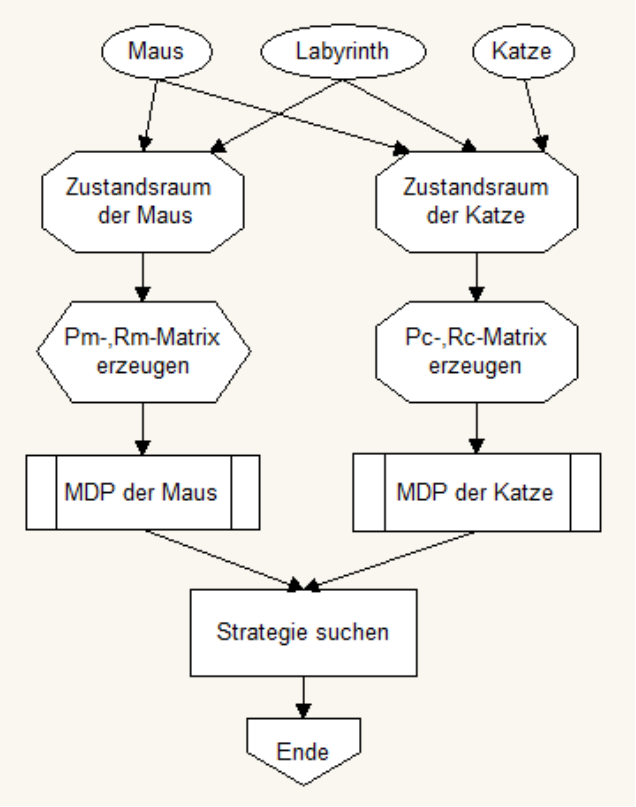

Abbildung 26: Struktur des blinden Maus Modells

Im neuen Modell sind die Zustandsräume der Maus und Katze getrennt. Die P-Matrix und R-Matrix der beiden Agenten wurden durch zwei getrennte Prozesse erzeugt. Diese Veränderung führt zu einer längeren Laufzeit des Systems. Die Struktur des Systems ist auch komplexer als die des vorherigen Modells.

4.2.2. Modell mit der blinden Maus

Zuerst wird das Modell mit einer blinden Maus erläutert. In diesem Modell gibt es keine Katze, nur ein Stück Käse. Durch das Modell kann das Bewegungsverhalten der Maus beobachtet werden. Das angewandte Labyrinth ist in der Abbildung 27 dargestellt. Es wird deutlich, dass es drei Bewegungsrichtungen der Maus gibt, die es ihr ermöglichen den Käse zu finden. Die kürzesten Wege zum Käse in den beiden Richtungen sind gleich lang (11 Schritte). Nun geht es um die Frage, in welche Richtung die Maus gehen wird.

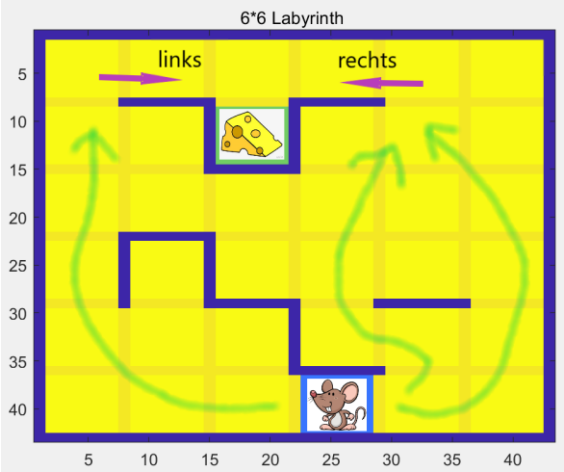

Abbildung 27: Mögliche Routen der Maus zum Käse

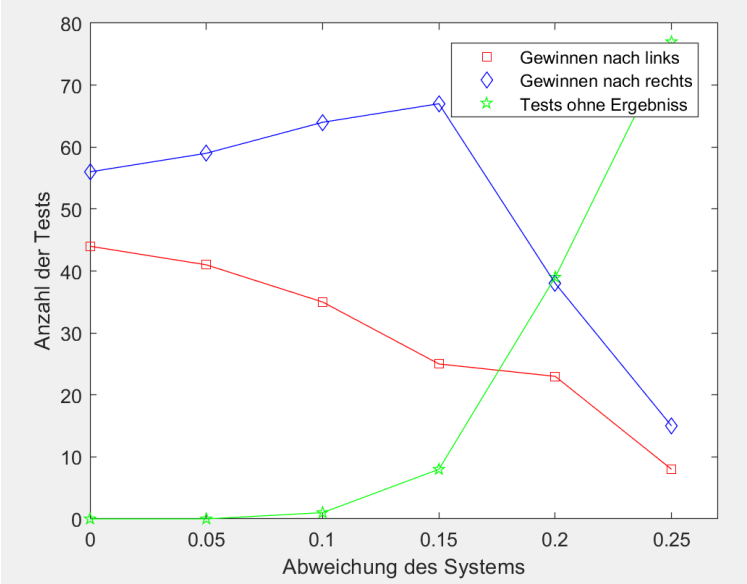

Diagramm 5: Bewegungsrichtungen des blinden Maus Modells

Das Diagramm 5 zeigt die Ergebnisse der Tests mit dem blinden Maus Modell. Der Test läuft 100 Mal ab. Die Abweichung des Systems steigt von 0 bis 25%. In der Abbildung kann man sehen, dass die Maus den Käse häufig nicht finden kann, wenn die Abweichung größer als 20% ist. Je höher die Abweichung des Systems, desto schwerer fällt es der Maus den Käse finden. Weil die Anzahl der Iterationen in diesen Tests nicht verändert wurde, haben viele Tests keine Ergebnisse. Wenn die Abweichung steigt, geht die Maus immer seltener in die linke Richtung. In der rechten Richtung steigt die Anzahl zuerst, wenn aber die Abweichung größer als 15% ist, fällt sie. Obwohl die Länge des kürzesten Wegs in den beiden Richtungen gleich ist, wählt die Maus die rechte Seite häufiger als linke Seite, weil es dort mehr Wände im Labyrinth gibt. Die Maus kann sich auf der linken Seite freier bewegen und hat dort größere Chancen, die Gefahr zu vermeiden.

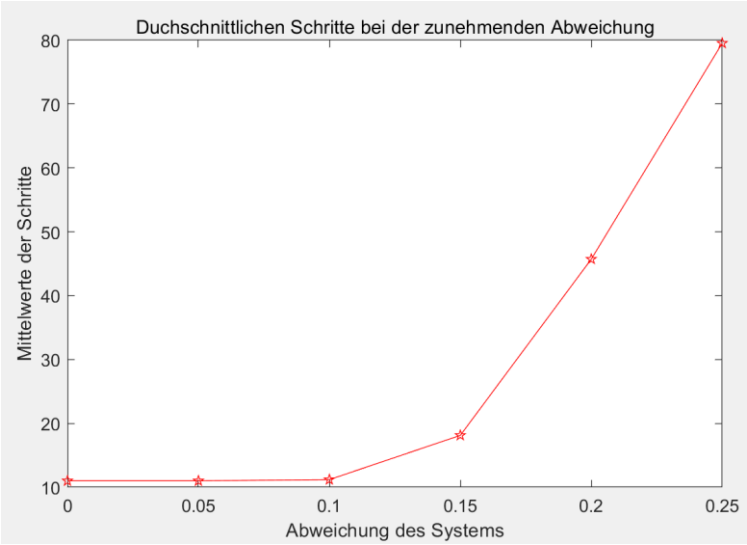

Diagramm 6: Durchschnittlichen Anzahl der Schritte des blinden Maus Modells

Das Diagramm 6 zeigt die durchschnittliche Anzahl der Schritte der Maus. Wenn die Abweichung größer als 15% ist, braucht die Maus mehr Schritte, um den Zielplatz zu erreichen. Aus der Analyse kann geschlossen werden, dass die 15% Abweichung des Systems eine Grenze darstellt. Wenn die Abweichung größer als 15% ist, kann die Maus häufig keinen optimalen Weg suchen. Wenn die Abweichung größer als 25% ist, gibt es oft keine Ergebnisse der Simulationen.

4.2.3. Modell mit der Katze und der blinden Maus

In diesem Kapitel wird das duale MDP-Modell mit der Katze und der blinden Maus in der unsicheren Umgebung diskutiert. Dabei wird auf zwei Modelle eingegangen: Das erste Modell hat nur eine Abweichung für die Maus, aber nicht für die Katze. Das zweite Modell hat Abweichungen für beide Agenten. Durch den Vergleich der zwei Modelle kann man die Eigenschaften der beiden Systeme besser auswerten. Der Einfluss der Abweichung kann auch klarer analysiert werden. Die Abweichung steigt von 0 bis 20% an. Der Test läuft 100 Mal ab.

Abbildung 28 zeigt den Startzustand des blinden Maus Modells. Es gibt zwei Plätze nahe dem Käse (Platz A und Platz B). Wenn die Maus durch den Platz A zum Käse geht, ist die die Richtung des Gewinnens links. Wenn die Maus durch den Platz B zum Käse geht, ist die Richtung des Gewinnens rechts. Die Katze befindet sich rechts oben des Labyrinths, sodass die rechte Richtung zum Käse gefährlicher für die Maus ist. Durch die Anwendung dieses Labyrinths werden die Richtungen des Gewinnens der Maus gut analysiert.

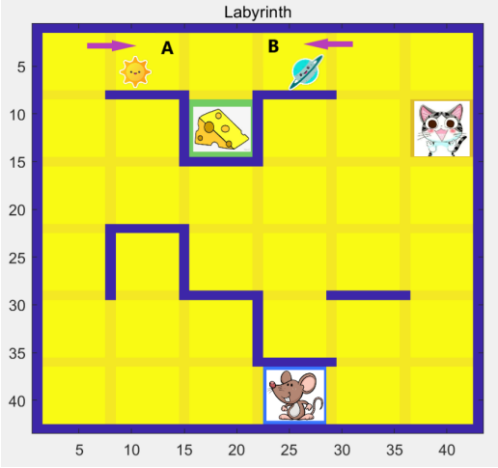

Abbildung 28: Startzustand des blinden Maus Modells

Das Diagramm 7 zeigt die Ergebnisse der Tests mit der Wahrscheinlichkeit des Gewinnens für die Maus und die Katze. Wenn die Abweichung der Maus oder beiden Agenten ansteigt, sinkt die Wahrscheinlichkeit des Gewinnens für die Maus auch sehr

schnell. Es gibt zwei Erklärungsmöglichkeiten. Eine Möglichkeit ist, dass die Maus eine gefährlichere Route wegen der zunehmenden Abweichung wählt. Eine andere Möglichkeit ist, dass die Maus mehr Schritte wegen der zunehmenden Abweichung benötigt, um den Käse zu finden. Mehr Schritte bedeuten längere Laufzeit und dies führt zu mehr Gefahren.

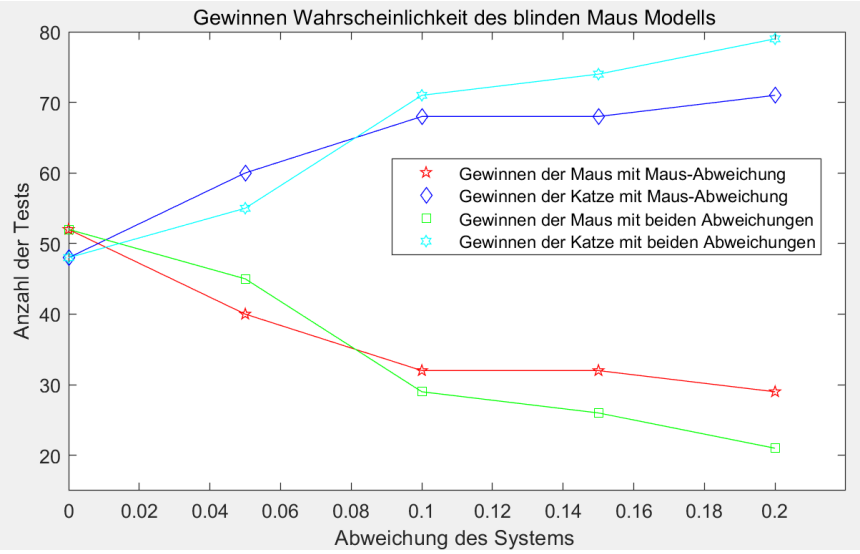

Diagramm 7: Gewinn-Wahrscheinlichkeit des blinden Maus Modells

Darüber hinaus wird deutlich, dass die Katze häufiger gewinnen kann, während die Abweichung der zwei Agenten ansteigt. Es verdeutlicht, dass die Abweichungen einen kleinen Einfluss auf die Katze haben. Die Kurven der beiden Modelle für die Maus oder die Katze sind ähnlich. Das bedeutet, dass die Abweichung der Katze die Eigenschaften des Systems wenig beeinflusst.

Diagramm 8 zeigt die Bewegungsrichtungen des blinden Maus Modells. Die Katze steht rechts oben im Labyrinth. Wenn die Maus eine linke Route wählt, hat die Maus mehr Chancen, die Simulation zu gewinnen. Wenn sie allerdings eine rechte Route wählt, ist die Gefahr größer, da sich die Katze dort befindet. Weil die Maus die Katze nicht beobachten kann, kann sie diese nicht aktiv vermeiden. Deshalb ist die rechte Seite für die blinde Maus gefährlicher als für die normale Maus.

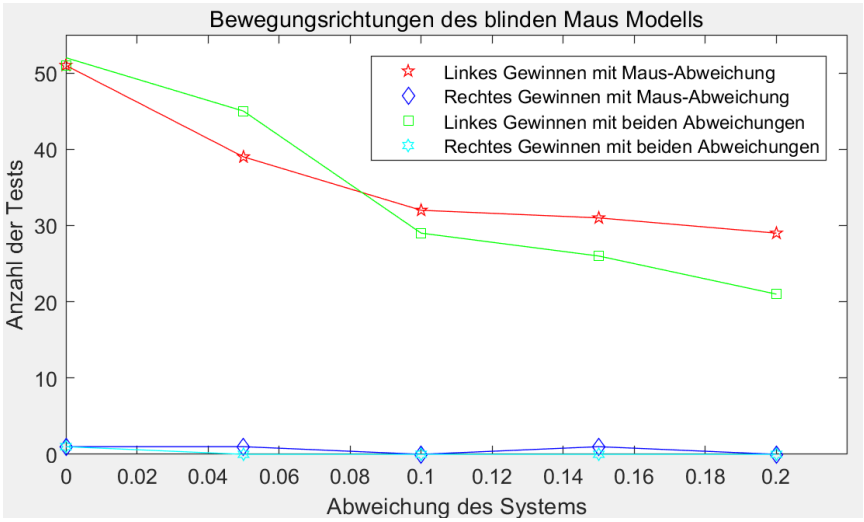

Diagramm 8: Bewegungsrichtungen des blinden Maus Modells

Dieses Diagramm zeigt, dass die Maus in beiden Modellen fast nie durch das Wählen der rechten Seite gewinnen kann. Wenn die Maus nach rechts geht, kann die Katze immer einen richtigen und kurzen Weg finden, um die Maus zu fangen. Weil die Anzahl des Gewinnens für die Maus wegen der ansteigenden Abweichung sinkt, sinkt die Anzahl des Gewinnens durch die Wahl der linken Seite.

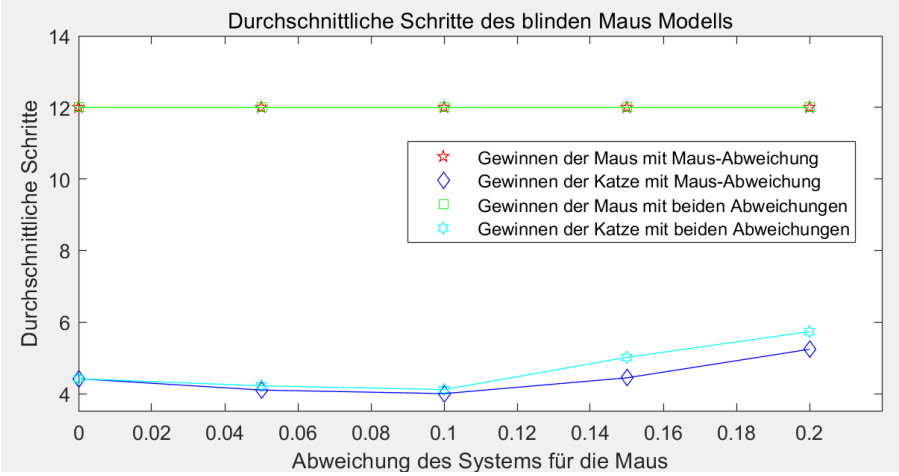

Diagramm 9: Durchschnittliche Anzahl der Schritte des blinden Maus Modells

Das Diagramm 9 zeigt die durchschnittliche Schrittanzahl der beiden Gewinner in den beiden Modellen. Die durchschnittliche Anzahl der Schritte der Maus beträgt in den beiden Modellen immer 12. Das bedeutet, dass die Maus nur gewinnen kann, wenn sie den richtigen und kürzesten Weg wählt. Die durchschnittliche Anzahl der Schritte

der Katze steigt langsamer beim Anstieg der Abweichung an, weil die Maus in einer falschen Richtung geht. Wenn die Maus einen Weg in der Gegenrichtung der Katze wählt, braucht die Katze mehr Schritte, um die Maus zu fangen.

Aus der oben durchgeführt Analyse kann gefolgert werden, dass die Abweichung der Maus einen starken Einfluss auf die Eigenschaften des Modells hat. Die Abweichung der Katze hat jedoch nur geringen Einfluss. Weil die Maus blind ist und sie deswegen die Katze nicht beobachten kann, wird die Maus immer von der Katze gefangen, wenn sie eine Route nahe der Katze wählt. Die zufälligen Bewegungen der Maus wegen der Abweichung des Systems können ihr nicht helfen, die Katze zu vermeiden.

4.3. Modell der blinden Maus mit einem speziellen Platz

4.3.1. Spezieller Platz im Labyrinth

Im vorherigen Modell ist die Abweichung des Systems für die beiden Agenten gleichmäßig verteilt. In der Realität gibt es jedoch die Möglichkeit, dass die Abweichungen des Systems nicht gleichmäßig verteilt ist. Um diese Situation zu simulieren, wird ein neues duales MDPs Modell mit einem speziellen Platz im Labyrinth aufgebaut.

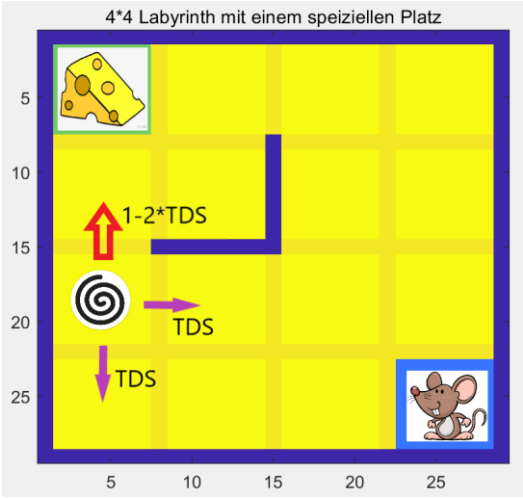

Abbildung 29: Spezieller Platz im Labyrinth

Abbildung 29 zeigt das Beispiel eines speziellen Platzes im Labyrinth. Die Koordination

des speziellen Platzes im Labyrinth ist (3,4). Es gibt eine besondere Abweichung TDS in diesem Platz. Wenn die Maus in diesem Platz steht, gibt es die spezielle Abweichung um (1-2*TDS) nach oben zu gehen. Es gibt noch eine Abweichung um TDS, in zwei andere Richtungen zu gehen. Das ganze Labyrinth hat eine Abweichung um TDM.

Nach dieser Veränderung des Modells wird die P- und R-Matrix der beiden Agenten geändert, aber die Struktur des Systems bleibt wie im blinden Maus Modell. Dies ist nur ein einfaches Beispiel. Theoretisch kann die spezielle Abweichung TDS in jedem Platz des Labyrinths und auch an mehreren Plätzen gleichzeitig existieren.

Wenn die Maus blind ist, kennt sie aber einen speziellen Platz nahe der Katze. Kann die Maus die Katze vermeiden?

4.3.2. Modell der blinden Maus mit einem speziellen Platz

Um den Einfluss des speziellen Platzes auf die Bewegungsrichtungen der Maus zu analysieren, wird ein diagonal symmetrisches Labyrinth mit nur einer Maus gebaut. Dieses Labyrinth wird in Abbildung 28 gezeigt. Die kürzesten Wege zum Käse für die Maus sind in beiden Richtungen (Osten des Käses und Süden des Käses) gleich lang. Dadurch wird der Einfluss des speziellen Platzes deutlich.

Das folgende Diagramm 10 zeigt die Ergebnisse der Tests zum Einfluss des speziellen Platzes. TDM ist die Abweichung des ganzen Labyrinths und TDS ist die Abweichung des speziellen Platzes. Die rote Kurve hat keine TDM Abweichung, die blaue Kurve hat eine TDM Abweichung von 1%.

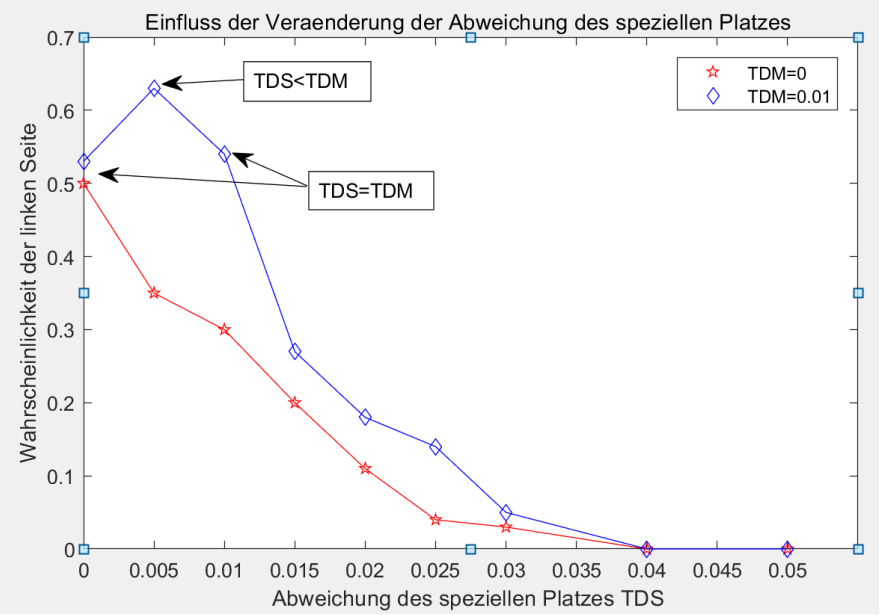

Diagramm 10: Veränderung der speziellen Abweichung

Wenn nur die Abweichung des speziellen Platzes vorhanden ist, sinkt die Kurve der Wahrscheinlichkeit der linken Seite bei der zunehmenden Abweichung. Die rote Kurve ist ziemlich linear, wenn die TDS Abweichung kleiner als 2.5% ist. Durch die Ergebnisse wird deutlich, dass die TDS Abweichung einen starken Einfluss der Richtung des Gewinnens der Maus hat. Wenn die TDS Abweichung größer als 4% ist, wählt die Maus niemals die linke Route zum Käse.

Zwischen den Anfängen der beiden Kurven besteht ein Unterschied, weil zu Beginn die TDM Abweichung größer als die TDS Abweichung ist. Wenn die TDS kleiner als die TDM ist, hat die TDM großen Einfluss auf die Ergebnisse. Deshalb wählt die Maus bei TDS=0.005 und TDM=0.01 (TDS<TDM) die rechte Seite häufiger als die linke Seite.

Wenn die TDM und die TDS Abweichungen gleich groß sind, wählt die Maus die beiden Richtungen ungefähr gleich häufig. In diesem Moment hat der spezielle Platz keinen Einfluss. Die blaue Kurve sinkt ähnlich stark wie die rote Kurve, wenn die TDS Abweichung größer als die TDM Abweichung ist.

Aus den beiden Kurven in Diagramm 10 wird ersichtlich, dass die TDS Abweichung sehr großen Einfluss auf die gewählte Bewegungsrichtung der Maus hat, wenn die TDS

Abweichung genug größer als die anderen Abweichungen ist. Wenn die Maus blind ist und das Labyrinth groß genug ist, kann die Maus die gefährliche Route vermeiden, falls ein spezieller Platz mit hoher Abweichung nahe der Katze liegt. Wenn das Labyrinth nicht groß genug ist, wird dieser Einfluss nicht deutlich.

4.3.3. Modell der dualen MDP Verfahren mit einem speziellen Platz

Die Analyse in Kapitel 4.3.2 zeigt, dass der spezielle Platz die Bewegungsrichtung der Maus beeinflussen kann. Kann der spezielle Platz im Modell der dualen MDP Verfahren auch funktionieren?

Abbildung 30: Startzustand des Modells mit einem speziellen Platz

Die Abbildung 30 zeigt den Startzustand des Modells mit einem speziellen Platz. Die Maus, die Katze und der Käse befinden sich in den drei Ecken des Labyrinths. Es gibt einen speziellen Platz nahe der Katze. Am Anfang der Simulation hat die Maus zwei Anfangsrichtungen (Norden (rechts) und Westen (links)). Es gibt auch zwei mögliche Wege zum Käse. Durch die Anwendung dieses Labyrinths kann man die Anfangsrichtungen und die Bewegungsrichtungen der Maus zum Käse analysieren. Zwei Modelle der dualen MDP Verfahren mit den verschiedenen Systemabweichungen werden simuliert, um die Beeinflussung der Systemabweichung zu analysieren.

Das Diagramm 11 zeigt die Veränderung der Wahrscheinlichkeit des Gewinnens, wenn

die Abweichung des speziellen Platzes von 0 bis 5% zunimmt. Die beiden Gruppen der Kurven haben einen ähnlichen Veränderungstrend. Hier wurde die TDS als die Abkürzung für die Abweichung des speziellen Platzes im Modell angewendet.

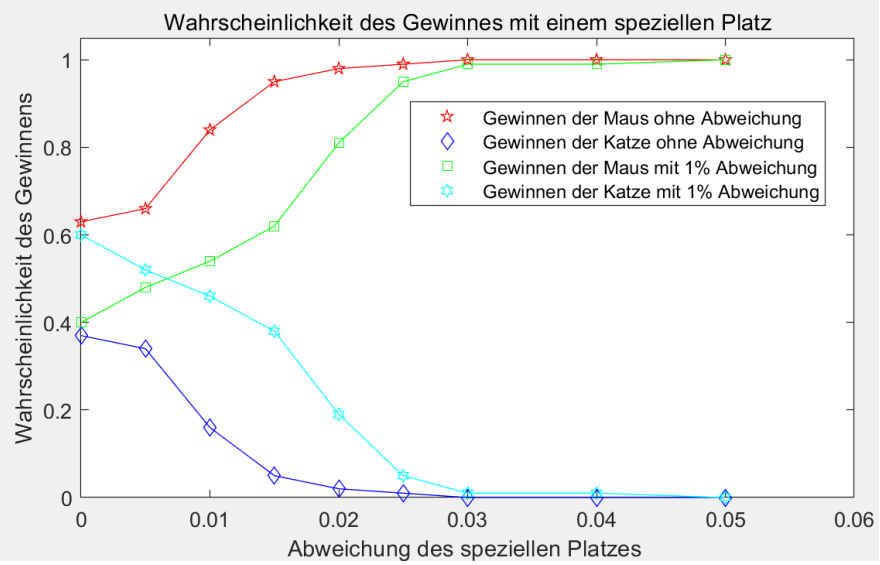

Diagramm 11: Wahrscheinlichkeit des Gewinnens mit einem speziellen Platz

Die roten und grünen Kurven steigen beim Anstieg der TDS. Wenn die TDS größer als 3% ist, kann die Maus zu fast 100% die Simulationen gewinnen. Wenn die TDS kleiner als 3% ist, gewinnt die Maus das Spiel zu einer geringeren Wahrscheinlichkeit, wenn das System eine Abweichung in allen Plätzen des Labyrinths hat. Es wird deutlich, dass die Maus gefährlicher ist, wenn es eine Systemabweichung im Modell gibt. Je höher die Systemabweichung ist, desto instabiler ist die Umgebung des Systems. Wenn die TDS höher als die Systemabweichung ist, hat die Maus besser Chancen, einen sicheren Weg zum Käse zu finden.

Das Diagramm 12 zeigt die Richtungen des Gewinnens der Maus. Es wird ersichtlich, dass die Maus bei der Wahl der linken Richtung kaum gewinnen kann. Dabei ist nicht relevant, ob es eine Systemabweichung gibt oder nicht. Die Maus kann das Spiel durch den Weg auf der rechten Seite häufiger gewinnen, wenn die TDS ansteigt. Dadurch zeigt sich, dass die TDS eine deutliche Funktion hat, um der Maus zu helfen, den rechten Weg zum Käse zu wählen und das Spiel zu gewinnen. Außerdem zeigt das Diagramm 12, dass der rechte Weg optimaler und sicherer ist.

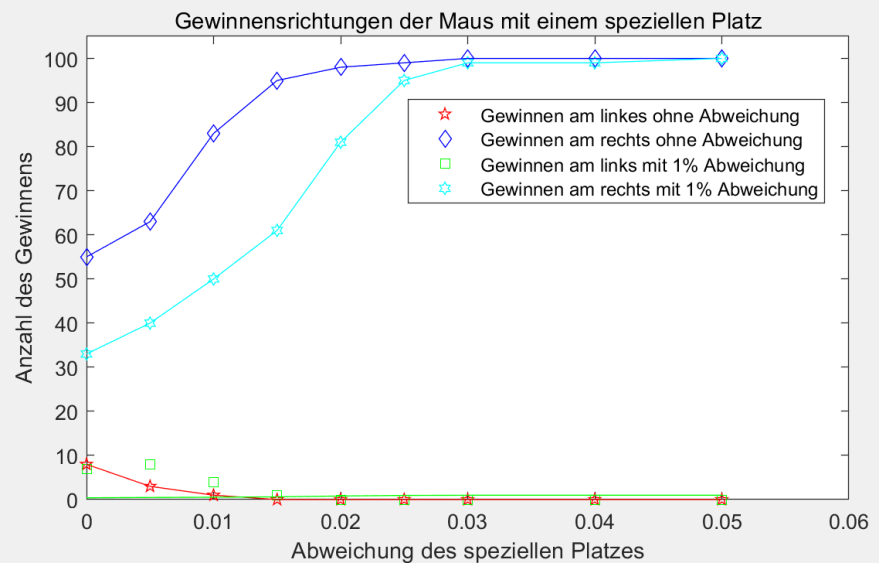

Diagramm 12: Richtungen des Gewinnens der Maus mit einem speziellen Platz

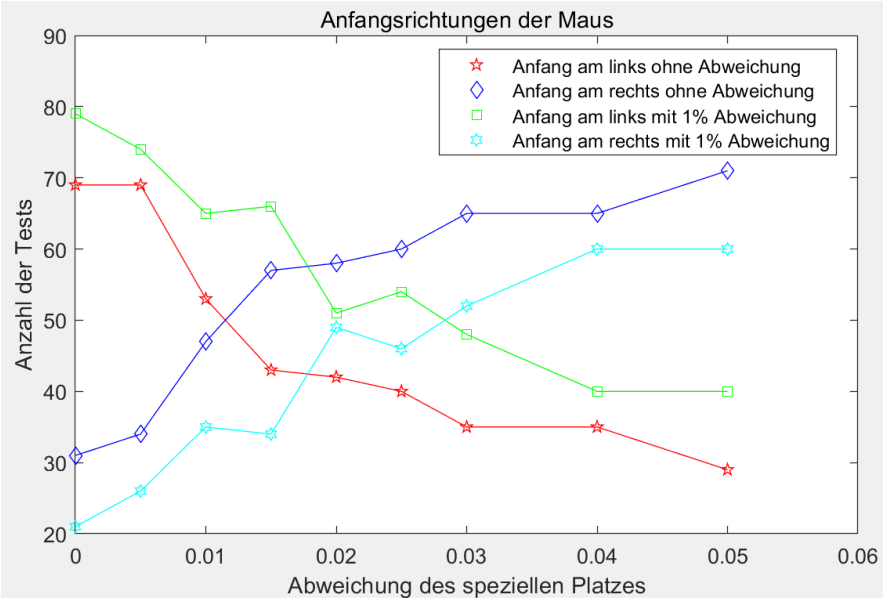

Diagramm 13: Anfangsrichtungen der Maus mit einem speziellen Platz

Diagramm 13 zeigt die Veränderung der Anfangsrichtungen der Maus beim Anstieg der TDS Abweichung. Wenn es keine Abweichungen gibt, wählt die Maus nach dem Lernen häufiger die linke Seite, weil die Maus die Katze nicht sehen kann und die linke Seite mehr Möglichkeiten bietet. Darüber hinaus wird deutlich, dass die linke Seite gefährlicher als die rechte ist. Wenn die TDS im Modell steigt, steigt die Anzahl der Anfangsrichtungen nach rechts ebenfalls an. Wenn die TDS groß genug ist, wählt die Maus zu Beginn die rechte Seite häufiger als die linke. Das Diagramm zeigt außerdem,

dass die Systemabweichung einen negativen Einfluss auf die Maus hat, weil der relative Wert der TDS wegen der Systemabweichung sinkt.

Der Vergleich der vorherigen drei Diagramme beweist, dass die Abweichung des speziellen Platzes, der sich nahe der Katze befindet, die gesuchte Strategie der Maus deutlich beeinflussen kann. Sie kann der Maus helfen, eine sichere Anfangsrichtung und einen sicheren Weg zum Käse zu wählen. Wenn die Maus einen sicheren Weg wählt, hat sie größere Chancen zu gewinnen. Der spezielle Platz hat die gleiche Funktion in der unsicheren Umgebung. Die TDS muss jedoch größer als die Systemabweichung sein. Wenn die TDS nicht genug groß ist, ist die Funktion des speziellen Platzes undeutlich.

In der Realität kann der spezielle Platz mit hoher Abweichung das System beeinflussen. Wenn sich der spezielle Platz nahe dem negativen Agenten befindet, kann er dem positiven blinden Agenten helfen, einen sicheren Weg zu finden. In anderen Modellen hat er vielleicht ähnliche Funktionen.

5. Zusammenfassung und Fazit

„Markov Decision Process"(MDP) ist ein mathematisches Modell, welches dazu dient Entscheidungsprobleme zu lösen. Der MDP Algorithmus wird oft in kognitiven Systemen angewendet. In der heutigen Zeit spielt das Multi-Agenten System eine wichtige Rolle. In dieser Arbeit wurden die Eigenschaften des dualen MDP-Modells analysiert. Das Maus-Katze-Spiel wurde moduliert, um dieses Modell zu simulieren.

Das duale MDP-Modell ist komplexer als das einzelne MDP-Modell. Die Veränderung der Umgebung kann die Aktion eines Agenten beeinflussen. Die Aktion des anderen Agenten kann ebenfalls die Aktion des ersten Agenten verändern. Im Maus-Katze-Spiel ist die Maus Gegenspieler zur Katze. Während des MDP Algorithmus kann der Agent durch die Zustände, die Aktionen und die Transitionen lernen und danach eine optimale Strategie finden.

Zuerst wurde das duale MDP-Modell in einer sicheren Umgebung simuliert. Zwei typische MDP Verfahren Wert-Iteration und Q-Lernen wurden in der Simulation angewendet. Die angewandten MDP Verfahren der beiden Agenten konnten gleich oder unterschiedlich sein. Die gefundenen Strategien zeigten, dass die Katze einen kurzen Weg zur Maus wählen kann. Die Maus kann die Katze umgehen und einen sicheren Weg zum Käse finden.

Die Analyse der Ergebnisse zeigte, dass die eingesetzten Belohnungen des Modells ein wichtiger Faktor sind. Die Belohnung r_{m+} $(r_{m+} > 0)$ stellt den Wunsch der Maus zum Käse dar. Wenn r_{m+} zunimmt, wählt die Maus zuerst den kurzen Weg zum Käse. Die Belohnung r_{m-} $(r_{m-} < 0)$ ist die Angst der Maus vor der Katze. Wenn r_{m-} absinkt, wählt die Maus zuerst den sicheren Weg. Der Vergleich der beide Belohnungen r_{m+} und r_{m-} stellte das Gewicht der beiden Faktoren für die Maus dar. Die Belohnung r_{k+} steht für den Wunsch der Katze die Maus zu fangen und r_{kk} ist die Beobachtung des Käses durch die Katze. Wenn $r_{k+} > r_{kk}$ ist, wählt die Katze zuerst eine

Bewegungsrichtung nahe zur Maus. Wenn $r_{k+} = r_{kk}$ ist, bewegt die Katze sich im Bereich in der Nähe des Käses. In dieser Situation ist die Maus gefährdet. Wenn $r_{k+} < r_{kk}$ ist, geht die Katze zuerst zum Käse: Dann wartet die Katze dort auf die Maus, sodass diese nur sehr geringe Chancen hat zu gewinnen.

Das duale MDP-Modell muss auch unter Unsicherheit arbeiten können. Aus diesem Grund wurde das normale duale MDP-Modell in unsicherer Umgebung simuliert. Der MDP Algorithmus TD-Learning wurde in unsicherer Umgebung angewendet. Durch die Simulation zeigte sich, dass die Abweichung des Systems die Bewegungsrichtung der Maus beeinflussen kann. Die gefundene Strategie der Simulation zeigt, dass die Maus auch dann einen sicheren Weg finden kann, wenn die Abweichung des Systems 10% ist. Wenn die Abweichung des Systems höher als 25% ist, ist es jedoch schwer, eine Strategie zu finden.

Anschließend wurde ein neues spezielles Modell in dieser Arbeit simuliert: das blinde Maus Modell. In diesem Modell ist die Maus blind und kann die Katze nicht sehen, was bedeutet, dass sie die Gefahr durch die Katze nicht kennen und vermeiden kann. In diesem Fall kann die Katze beim Anstieg der Abweichung mehrmals gewinnen. Die Abweichung des Systems hat einen großen Einfluss auf die Maus. Wenn die blinde Maus die Bewegungsrichtung nahe der Katze wählt, hat sie nur geringe Chancen zu gewinnen. Wählt sie jedoch den kurzen Weg in der sicheren Bewegungsrichtung, gewinnt die Maus am Ende.

Ein spezieller Platz wurde dem blinden Maus Modell hinzugefügt, um der Maus zu helfen, die Gefahr durch die Katze zu vermeiden. Dieser spezielle Platz befindet sich nahe der Katze und er hat eine spezielle Abweichung TDS. Wenn die Abweichung TDS größer als die Abweichung des Systems ist, wählt die Maus die Bewegungsrichtung nahe der Katze seltener. Je höher der Differenzwert ist, desto stärker ist der Einfluss des speziellen Platzes. Ist die Abweichung TDS 5% größer als die Abweichung des Systems, geht die Maus immer in die sichere Richtung.

Die angewandten Labyrinthe in dieser Arbeit sind kleiner als 6*6, weil die Laufzeit der Simulation in unsicherer Umgebung lange dauert. Die Ergebnisse des Tests in einem großen Labyrinth kann man in der zukünftigen Arbeit noch beobachten. Die angewandten MDP Algorithmen in dieser Arbeit sind Wert-Iteration, Q-Lernen und TD-Learning. Ein erweiterter Algorithmus für ein spezielles Modell könnte in einem nächsten Schritt entwickelt werden.

Literaturverzeichnis

[Mat17] Matthias Wolff: Vorlesungsskript Kognitive Systeme 1-Perzeption und Aktion, 2017/18

[Mey15] Werner Meyer: Modellierung und Implementierung einer kognitiven Verhaltenssteuerung, 2015
URL: https://www-docs.b-tu.de/fg-kommunikationstechnik/public/Abschlussarbeiten/meyer.pdf

[Tob15] Tobias Starzak: Kognition bei Menschen und Tieren, eine vergleichende philosophische Perspektive, 2015
ISBN: 978-3-11-037477-3

[Cla50] Theseus Maze, MIT Museum, Cambridge, Massachusetts
URL: https://de.wikipedia.org/wiki/Claude_Shannon

[Wal13] Waldmann, K.-H.; Stocker, U.M.: Stochastische Modelle - Eine anwendungsorientierte Einführung, 2013, VIII; 263 S. 46 Abb., Softcover
ISBN: 978-3-642-32911-1

[Wol09] Ertel, Wolfgang: Grundkurs Künstliche Intelligenz, 2. Überarbeitete Auflage 2009, Vieweg+Teubner Verlag
ISBN: 978-3-8348-0783-0

[Joh05] Johannes Knabe: Koorperatives Reinforcement Lernen in Multiagenten-systemen, 2005
URL: http://panmental.de/papers/CooperativeRLinMAS.pdf

[Cra96] Craig Boutilier: Planning, Learning and Coordination in Multiagent Decision Processes, 1996, 96 Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge, Seiten 195-210
ISBN: 1-55860-417-9
URL: https://www.cs.toronto.edu/~cebly/Papers/tark96.pdf

[Jea14] Jean-Francois Raskin, Ocan Sankur: Multiple-Environment Markov Decision Processes; 2014

URL: https://arxiv.org/abs/1405.4733

[Nic11] Nicole Bäuerle, Ulrich Rieder: Markov Decision Processes with Applications to Finance, 2011
ISBN: 978-3-642-18323-2

[Sha17] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, John Vian: Deep Decentralized Multi-task Multi-Agent Reinforcement Learning under Partial Observability, 2017, Proceedings of the 34 th International Conference on Machine Learning

[Sne01] Snehasis Mukhopadhy a yand Bindu Jain: Multi-Agent Marko vDecision Processes With Limited Agent Communication, 2001, Proceedings of the 2001 IEEE International Symposium on intelligent Control

[Jor15] Joris Scharpff, Diederik M. Roijers, Frans A. Oliehoek, Matthijs T. J. Spaan, Mathijs M. de Weerdt: Solving Multi-agent MDPs Optimally with Conditional Return Graphs, 2015, The 10th Annual Workshop on Multiagent Sequential Decision-Making Under Uncertainty (MSDM-2015)

Abkürzung

MDP	Markow-Entscheidungsprozess oder „Markov Decision Process"
MMDP	Multiagenten-System
M	Maus
K	Katze
WI	Wert-Iteration oder Value-Iteration
QL	Q-Lernen oder Q-Learning
TD	TD-Learning
TDM	Abweichung des Systems der Maus
TDK	Abweichung des Systems der Katze
TDS	Abweichung des speziellen Platzes