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

# Modellierung und Implementierung einer kognitiven Verhaltenssteuerung

Eingereicht von:

Name: Werner Meyer
Matr.-Nr.: 3215240
Anschrift: Am Kniebusch 22
16303 Schwedt/Oder

Betreut von:

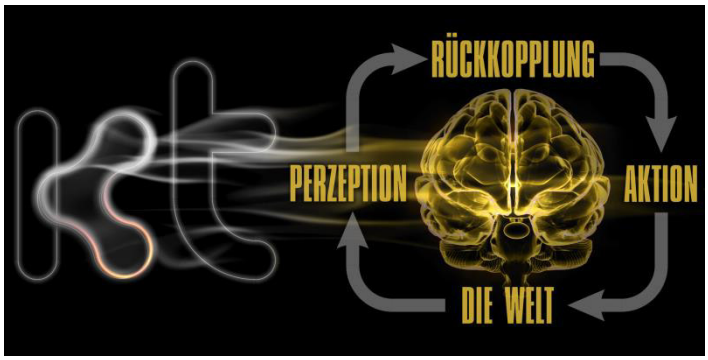
Dr. -Ing. Ronald Römer

Cottbus, 05.01.2015

THEMENANGEBOT FÜR EINE

# MASTERARBEIT

Lehrstuhl
Kommunikationstechnik




Thema:

## Modellierung und Implementierung einer kognitiven Verhaltenssteuerung

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 die Minimierung der Gesamtkosten 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, **P**artially-**O**bservable-**MDP**).

In dieser Arbeit sollen in einem ersten Schritt das Strategietraining unter Verwendung des MDP-Verfahrens implementiert und am Beispiel einer klassischen Objektsteuerung in einer Labyrinth-Umgebung demonstriert und dessen Adaptionsfähigkeit an eine veränderte Umgebung nachgewiesen werden. In einem nächsten Schritt soll die Objektsteuerung aus kognitiver Sicht betrachtet werden und um eine semantische Ebene erweitert werden. Erst mit dieser Erweiterung können höhere psychologische Wirkprinzipien wie „Coping“ in ein Kognitives System eingebunden werden.

Betreuer:

Dr.-Ing. Ronald Römer
T: 0355 695007
E: name@tu-cottbus.de

## Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich

- die vorliegende Masterarbeit selbständig und ohne unerlaubte Hilfe angefertigt habe,
- andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und
- die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Cottbus, 05.01.2015 _____

## Danksagung

An dieser Stelle möchte ich einen ganz besonderen Dank meinem Betreuer, Herrn Dr.-Ing. Römer, aussprechen, dem ich eine sehr gute Betreuung zu verdanken habe. Die regelmäßigen Konsultationen, unter anderem auch mit Herrn Prof. Dr.-Ing. habil. Wolff, brachten mir immer wieder neue Inspirationen und Ansätze zur Bewältigung der Aufgabe.
Auch den anderen Mitarbeitern aus der Fakultät Kommunikationstechnik möchte ich für die Aufmerksamkeit danken, die sie mir während meines Zwischenvortrages gaben, wodurch ebenfalls eine interessante Diskussion zu dem Thema entstand.

Desweiteren möchte ich mich auch bei meinen Eltern und meinem Bruder recht herzlich bedanken, die mir über das gesamte Studium finanziell und auch emotional zur Seite standen.

## Kurzfassung

In dieser Arbeit wird eine Verhaltenssteuerung, basierend auf dem Verfahren des bewertenden Lernens, in ein Kognitives System implementiert. Als Bezug wird Shannon's Maus Theseus verwendet, die das Ziel hat sich in einem Labyrinth zu orientieren. Die Modellierung erfolgte mit dem Markov-Entscheidungsprozess, was in einer Referenzapplikation umgesetzt wurde. Dabei wurden drei Verfahren des bewertenden Lernens betrachtet. Als eine Erweiterung der Wert-Iteration wurde das Q-Lernen eingesetzt, mit dessen Algorithmus eine Strategie aus dem angeeigneten Wissen über die Umgebung gebildet werden kann, ohne die Umgebung erneut mit einzubeziehen. Um unvorhergesehene Einflüsse aus der Umwelt mit einzuordnen, wurde das Q-Lernen auf das Temporal-Differenz-Lernen erweitert.

Die Anpassung an eine veränderte Umwelt konnte mit Hilfe dieser Verfahren dargestellt werden. Der Nachteil war, dass eine Anpassung des Aktionsrepertoires nicht erfolgen konnte, wenn die Umgebung dies erforderte. Um dieses Problem zu lösen, wird der Aspekt des Bewältigungsverhaltens aus der Psychologie mit einbezogen. Es wird ein Coping-Algorithmus präsentiert, der auf einer semantischen Modellierung beruht und damit tatsächlich in den Geltungsbereich Kognitiver Systeme fällt. Es wird gezeigt, dass dieser Algorithmus den klassischen Ansätzen überlegen ist, da hier das Aktionsrepertoire von der Umgebung geprägt wird. Ein wichtiger Vorteil besteht darin, dass der MDP-Mechanismus weiterhin verwendet werden kann.

## Abstract

In this disquisition it will be implemented a behavior control into a cognitive system based on the procedure of Reinforcement Learning. Shannon's mouse Theseus is used as a reference. In that thesis the mouse has to orientate oneself in a Labyrinth. The modeling is affected by using the Markov-Decision-Process, which is influenced by the reference application. In doing so there will be considered three methods of Reinforcement Learning. In addition to the Value-Iteration it is used the algorithm of Q-Learning. By using that algorithm it is possible to create a strategy of the environment without involving the environment again. The Q-Learning system is upgraded by using the temporal-difference-learning for integrate unpredictable environmental influences.

There by it is feasible to illustrate the adaptation of the changing environmental influences. But it is impossible to involve an adjustment of the action repertoires. To solve that problem it is implemented the aspect of the coping behavior as a field of the psychology. A coping-algorithm is presented which based on a semantic modeling. Because of that it can classified as a part of a cognitive system. It can be shown that this algorithm is better than the classic theories because the actions repertoire is influenced by its environment. Another important advantage is that the MDP-Mechanism can be used continuously.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis



# Diagrammverzeichnis

## Formelgrößen

| | |
|---|---|
| $a$ | Gewählte Aktion |
| $a' = a_{t+1}$ | Nachfolgeaktion |
| $s$ | Aktueller Zustand |
| $z' = z_{t+1}$ | Nachfolgezustand |
| $\delta(z_t, a_t) = z_{t+1}$ | Übergangsfunktion |
| $r_t = r(z_t, a_t)$ | Direkte Belohnung |
| $\pi$ | Lösungsstrategie |
| $\pi^*$ | Optimale Lösungsstrategie |
| $V(z)$ | Wert / Abgeschwächte Belohnung, (Belohnung aus direkter und abgeschwächter Folgebelohnung) |
| $V^{\pi^*}=V^*$ | Optimaler Wert / optimale abgeschwächte Belohnung |
| $\gamma$ | Diskontierungsfaktor |
| $Q(z, a)$ | Q-Matrix , Wert / abgeschwächte Belohnung für Q-Lernen |
| $N$ | Gesamtzahl an Iterationen |
| $n$ | Gegenwärtige Iteration |
| $b_n(z, a)$ | Zugeordnete Anzahl verwendeter Transitionen in Abhängigkeit von Zustand und Aktion |

# 1 Einführung und Motivation

Mit voranschreitender Computertechnik gewann die Thematik der Künstlichen Intelligenz in der Forschung zunehmend an Bedeutung. Der Begriff „Künstliche Intelligenz“ wurde 1955 erstmals erwähnt und ein Jahr später von Claude Shannon, McCarthy, Minsky und Nathaniel Rochester geprägt. Das Forschungsgebiet der Künstlichen Intelligenz war geboren [The].

Die Definition von „Künstliche Intelligenz“ ist gegenwärtig nicht eindeutig. Dies liegt daran, dass der Teilbegriff „Intelligenz“ wissenschaftlich nicht klar abgegrenzt ist. Forscher aus unterschiedlichen Themengebieten, wie Beispielsweise der Psychologie, Biologie, Elektrotechnik, Informatik, Kommunikationstechnik und Mathematik beschäftigen sich mit diesem Thema.
Die Untersuchung des menschlichen Gehirns erfolgt Beispielsweise auf den Wegen der Introspektion, durch psychologische Experimente und über die Hirntomografie, die dazu beitragen sollen die Funktionsweise des Gehirns zu verstehen [RN12].
Ein System muss nicht unbedingt nach dem Schema eines menschlichen Gehirns funktionieren um künstliche Intelligenz aufzuweisen. Wird das Lösen einer Mathematikaufgabe als Intelligent betrachtet, so sind die Computer von heute bereits jedem menschlichen Kopfrechnen überlegen. Dem gegenüber ist bei der Interpretation zur Lösung einer Aufgabe, wie beispielsweise $0^0$ dem Rechner lediglich eine vordefinierte Lösung von 1, nicht definierbar oder Error vorbehalten [Beu14].

Es ist nicht klar einzugrenzen, welche Bestandteile zu der Fähigkeit Intelligenz gehören. Kognitive Fähigkeiten werden dennoch von allen Experten als ein wichtiger Bestandteil von Intelligenz vorausgesetzt. Darunter zählen Fähigkeiten, wie Wahrnehmung, Gedächtnis, Lernen, Denkvermögen, logisches Schließen und einige Weitere [SF].

Das Umsetzen kognitiver Fähigkeiten in einem technischen System spielt in dieser Arbeit eine wesentliche Rolle. Dazu zählt das Lernen und Orientieren in einer Umge-

bung sowie das Anwenden einer Copingstrategie zur Bewältigung eines auftretenden Problems.

Die Fähigkeit des Lernens ist in der Verhaltenssteuerung mit am bedeutendsten. Claude Shannon demonstrierte bereits 1952 “Theseus”, eine technische Maus die von einer Maschine durch ein Labyrinth gesteuert wurde. Theseus musste erst lernen sich in dem Labyrinth zu orientieren, um den richtigen Weg zu finden. Im Anschluss der Lernphase fand die Maus den direkten Weg. Erst nach einer Umstellung des Labyrinths startete Theseus zwar direkt, aber an einer Sackgasse angekommen begann sie wieder von vorn einen neuen Weg zu erlernen [Hog].

Das Labyrinth in der sich die technische Maus befindet ist kartesisch angeordnet. In Abbildung 1 wird die Aufzeichnung des Lernvorgangs der Maus dargestellt. Mit Hilfe von Langzeitbelichtung und einer Lampe auf dem Rücken der Maus entstand diese Abbildung. Die Wände sind hier nicht mit eingeblendet, dennoch ist aus dem ersten Teilbild (oben links) ersichtlich, welche Struktur das Labyrinth besitzt. Die Zielposition befindet sich unten rechts.

Im ersten Teilbild (1) wird der Lernfortschritt von Theseus dargestellt. Kurz nach dem Start ist zu erkennen, dass die Maus mehrmals in eine Sackgasse gelaufen ist. Dort drehte sie sich in alle Richtungen, um einen weiteren Weg zu finden. Der einzig vorhandene Weg war jedoch zurück über die Strecke, die sie zuvor bereits abgelaufen ist, bis zu einem Status, in dem sie eine neue Strecke betreten kann.

Im zweiten Teilbild (2) werden die Lösung und damit der optimale Weg zum Ziel dargestellt. Theseus wurde, nun anscheinend ausgestattet mit dem gelernten Wissen über das Labyrinth, an der Startposition ausgesetzt und kannte sofort den zu beschreitenden Weg.

Im darauffolgenden Teilbild (3) ist kein wirklicher Unterschied zum zweiten Teilbild vorhanden. Theseus wird lediglich an einem anderen Startpunkt ausgesetzt. Dieser ist ihr allerdings bereits aus der Lernphase bekannt. Das Labyrinth wurde nicht verändert und die Maus kann ihr vorheriges Wissen über das Labyrinth anwenden.

Im letzten Bildabschnitt (4) wird Theseus in einem ihm noch nicht bekannten Abschnitt des Labyrinths ausgesetzt. Zu Beginn setzt die Lernphase ein, da Theseus die neue Umgebung erst kennen lernen muss. Sobald die Maus aber auf die ihr bekannte Route trifft, verfolgt sie diese bis zum Ziel. Wäre im bekannten Abschnitt ebenfalls eine Änderung vorgenommen worden, würde die Maus an der ersten veränderten Stelle mit einer erneuten Lernphase beginnen, bis sie das Ziel erreicht hat.

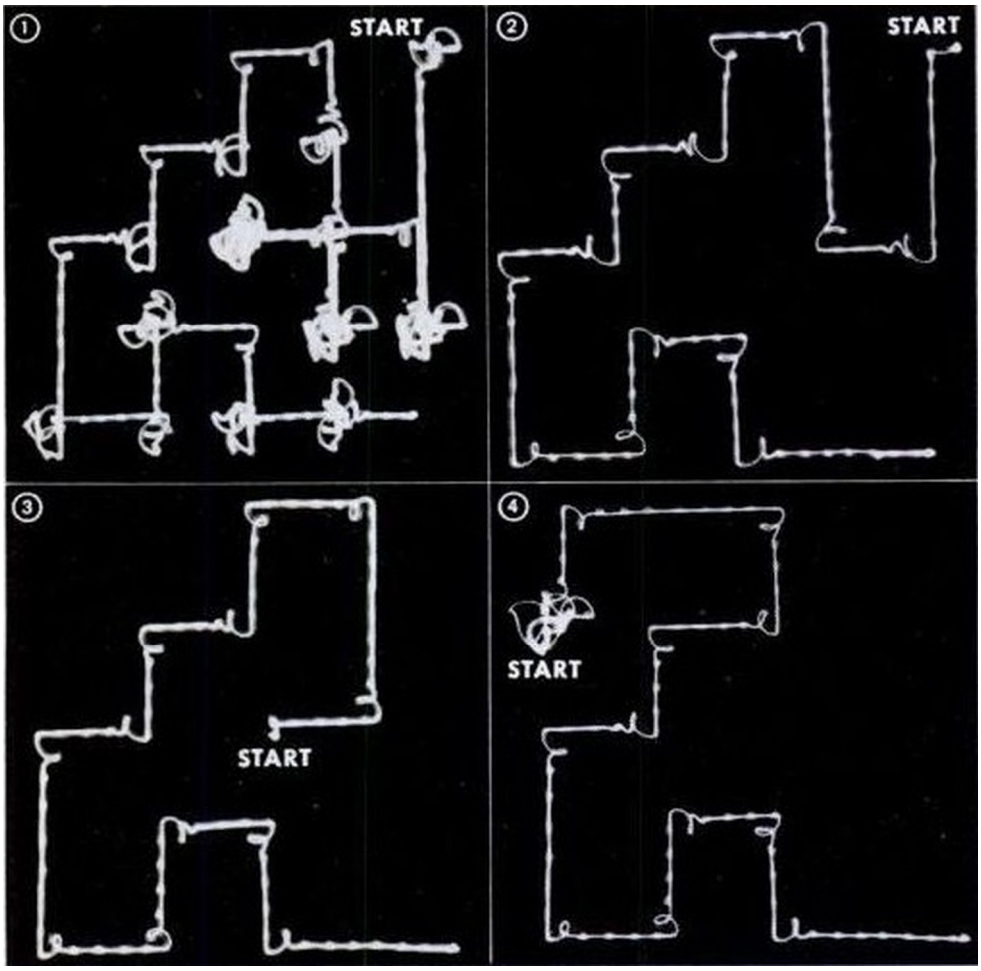



**Abbildung 1: Memory Tests - Theseus Lernfortschritte [Hog]**

Die Maus Theseus machte den Eindruck, als sammelte sie während ihrer Erkundungstour Erfahrungen, um das Ziel im Labyrinth zu finden. Sie lernte dazu, strebt nach einer Lösung und bleibt nicht ahnungslos stehen, wenn sie sich in einem ihr nicht bekannten Zustand befindet. Somit erschuf Shannon ein System, das Kognitive Fähigkeiten aufweist [Hog].

War Theseus jedoch eingesperrt und konnte das Ziel nicht erreichen, so hörte die Maus nicht auf nach einem passenden Weg zu suchen. Sie passte sich der Umgebung nicht an, sondern hielt an ihrer aktuellen Einstellung fest. Theseus war somit kognitiv beschränkt und besaß demgegenüber keine Adaptionsfähigkeit.

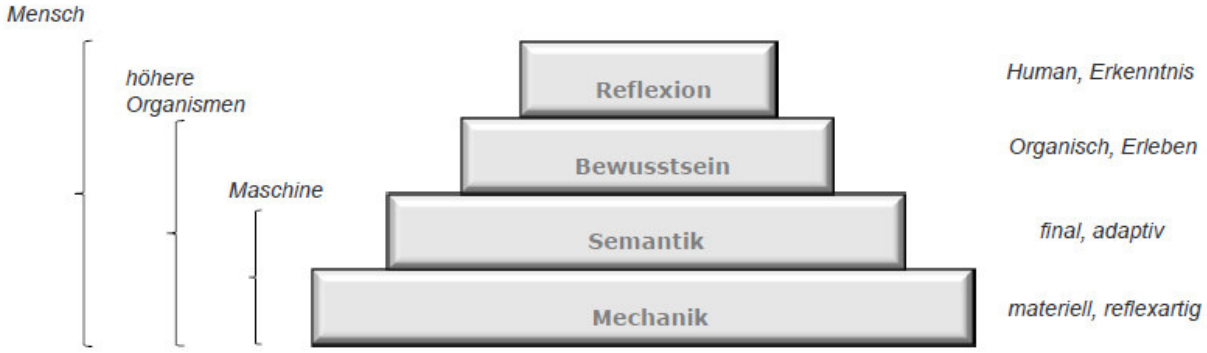



**Abbildung 2: Entwicklungspyramide nach N.Bischof [Röm14]**

Um ein finales System zu erschaffen, fehlt die Erweiterung um die Ebene der Semantik. Die Entwicklungspyramide von N. Bischof stellt dar, dass eine Maschine aus der Semantik und der Mechanik bestehen kann (Abbildung 2). Shannon's Maus Theseus war jedoch rein elektromechanisch.

Die Maus besitzt keine eigene Vorstellung zu den einzelnen Bewegungsmöglichkeiten, denn ihr Handlungsrepertoire wird durch Relais, die sich unterhalb des Labyrinths befinden, gesteuert. Theseus besitzt demnach kein eigenständiges Wissen über das Labyrinth. Vielmehr hinterlässt die Maus auf jedem Feld eine Notiz (Relais), in welche Richtung sie zuletzt gelaufen ist und ändert diese wiederum ab, sobald sie aus dieser Richtung zurückkehrt.
Erneut ausgesetzt auf einem bereits besuchten Feld folgt Theseus lediglich den Notizen, bis sie das Ziel erreicht oder auf noch unbekannte Felder trifft.

Das Verhalten von Theseus soll zuerst mit Hilfe des bewertenden Lernens (Reinforcement-Learning) nachgebildet werden. Dabei ist folgender Systementwurf aus Abbildung 3 zu betrachten.
Das Objekt stellt das Labyrinth in dessen aktuellem Zustand dar. Über die Perzeption wird die Wahrnehmung für den aktuellen Zustand des Kognitiven Systems realisiert. Im oberen Teil erfolgt die Anwendung der Verhaltenssteuerung, bei der hier auf das MDP-Model (Markov Decision Process) zurückgegriffen wird.

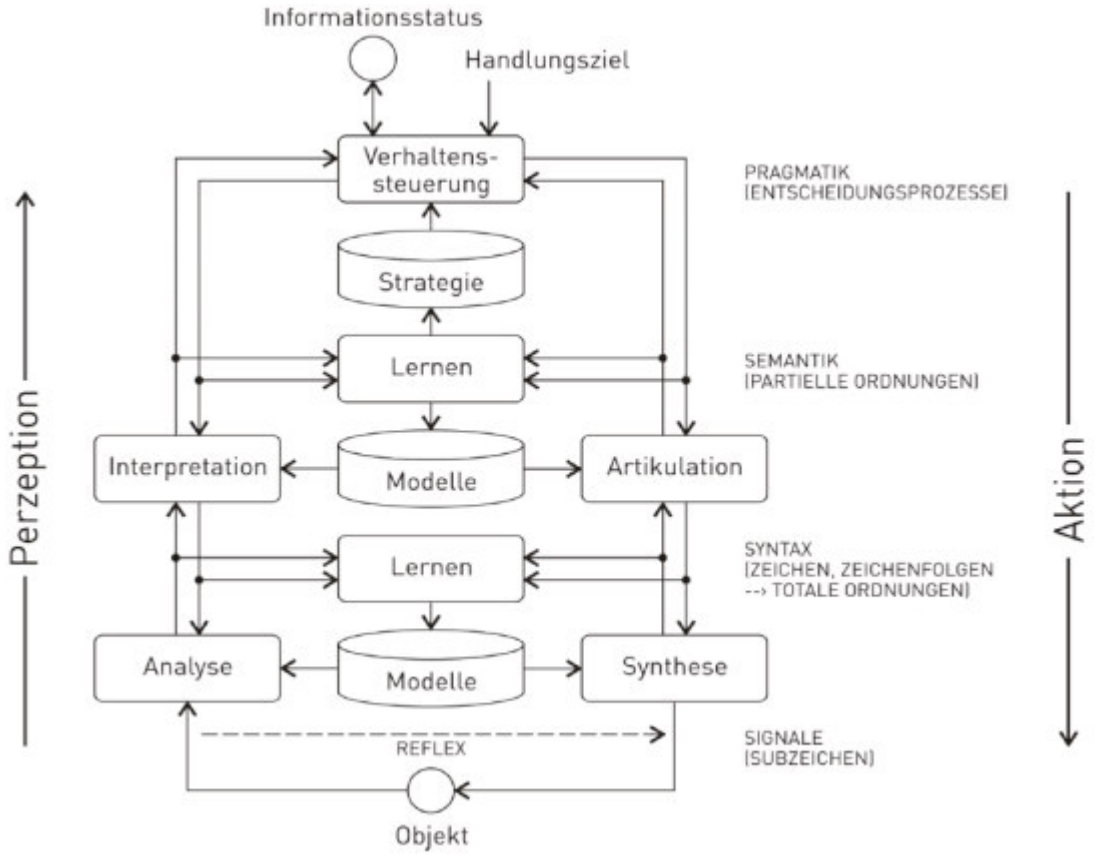



**Abbildung 3: kognitiver Systementwurf [Wol]**

Mit der Verhaltenssteuerung soll eine Strategie gelernt werden, deren Ausführung die Aktion in diesem Systementwurf darstellt und somit für eine Veränderung im Labyrinth und damit auch des Objektes sorgt.

In dieser Arbeit soll eine Kognitive Verhaltenssteuerung für Theseus entwickelt werden. Zunächst werden in Kapitel 2 die wichtigen Verfahren für eine deterministische und nichtdeterministische Umgebung dargestellt. In Kapitel 3 wird eine Referenzapplikation beschrieben, mit der sich Theseus an eine veränderte Umgebung anpassen kann. Die Referenzapplikation erfolgt im Rahmen der Möglichkeiten der MDP-Modellierung. In Kapitel 4 wird schließlich ein Algorithmus zum Bewältigungsverhalten entwickelt, der eine semantische Modellierung einbezieht und über die Möglichkeiten des MDP-Modells hinausgeht. Eine Zusammenfassung und ein Ausblick auf zukünftige Arbeiten schließen die Arbeit ab.

# 2 Grundlagen

Auch das Lernen muss gelernt sein. Damit ein Kognitives System lernen kann, muss die Form des Lernens in diesem System implementiert sein. Dabei werden unterschiedliche Methoden eingesetzt. Zum Beispiel wird bei künstlichen Neuronalen Netzen das Lernen, bei dem eine umfangreiche Datenmenge verwendet wird, durch Lehrer angewendet, um dem System ein gewünschtes Verhalten beizubringen.
Anhand der Eingangsdaten und den dazu passenden Ausgabedaten kann dem System das gewünschte Verhalten beigebracht werden. Das System lernt durch die Übungsdaten, wie es sich zu verhalten hat. Je umfangreicher diese Daten sind, desto genauer arbeitet dieses System im Nachhinein. Jedoch stehen diese Datenmengen nicht immer zur Verfügung und somit werden auch andere Techniken des Lernens benötigt.

Bei der hier gestellten Aufgabe, der Lösung des Weges in einem Labyrinth, gibt es keine Eingabedaten, denen passende Ausgabedaten zugeordnet werden können.

Ein Labyrinth kann auf unterschiedliche Weise gelöst werden. Zum Beispiel durch die Rechte-Hand-Regel, bei der immer einer Wandseite gefolgt wird oder dem komplizierteren Pledge-Algorithmus, bei dem zusätzlich der Winkel zur momentanen Ausrichtung zur Startposition in Betracht gezogen wird.
Ziel ist es aber mit Hilfe des Markov-Entscheidungsprozesses ein Verfahren zu schaffen, bei dem kein vorgegebener Algorithmus das Labyrinth löst, sondern vielmehr Wissen über die Struktur des Labyrinths angeeignet wird und das dieses gelernte Wissen zur Lösung des Labyrinths wiederholt angewendet werden kann.

Der Markov – Entscheidungsprozess ist ein Verfahren des verstärkenden Lernens (engl. Reinforcement Learning), das auf einem Belohnungssystem beruht. Dabei trifft das System in der Lernphase zufällige oder nach einem Schema vorgegebene Entscheidungen. Fallen diese Entscheidungen positiv aus, so wird das System belohnt. Bei negativen Entscheidungen wird das System bestraft (negative Belohnung).

Anhand der folgenden zwei Beispiele soll der Markov – Entscheidungsprozess sowie die dazugehörigen Schemen der Wert-Iteration und das Q-Learning erklärt werden.

## 2.1 Beispiel 1: Der Laufroboter

Bei dem Laufroboter wird der Markov-Entscheidungsprozess auf einen einarmigen Roboter (Agent) angewendet, der das Ziel hat sich mit Hilfe dieses Armes auf einer ebenen Fläche vorwärts zu bewegen. Dies wäre mit einem Bagger vergleichbar, der keine Räder besitzt und versucht mit Hilfe des Schaufelarmes sich fortzubewegen (Abbildung 4) [Ert09].

Der Agent ist hier der Roboter. Er stellt das Objekt dar, das den Algorithmus des Reinforcement Learning anwenden soll.
Der Arm des Roboters besteht aus zwei Gelenken. Die Spitze davon kann diskrete Werte in Y- und X- Richtung annehmen. Die Koordinaten geben den Zustand in dem sich der Roboterarm befindet an.

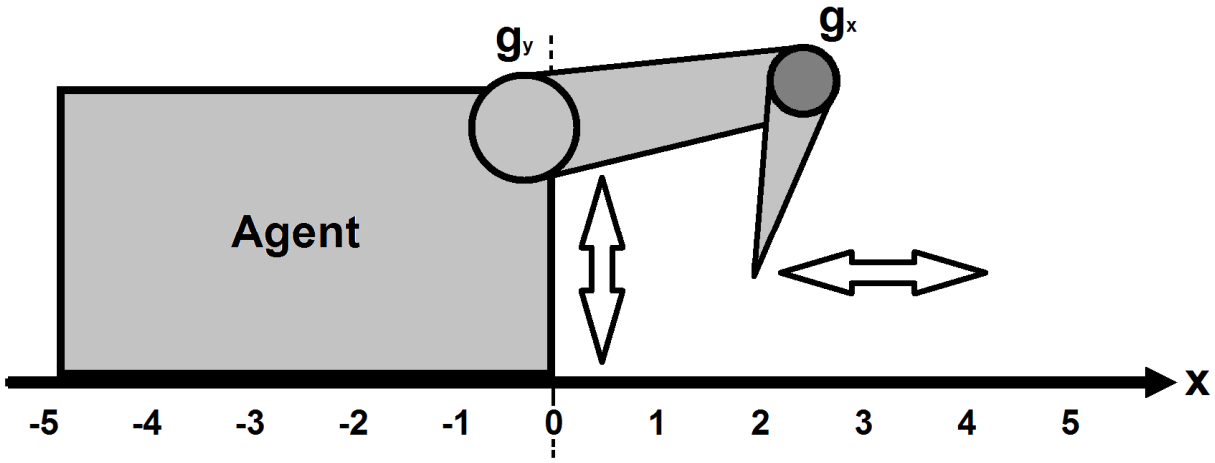


Abbildung 4: Laufroboter [Ert09]

Der Raum in dem sich die Spitze des Roboterarmes befinden kann, besteht aus einer endlichen Anzahl von Möglichkeiten. Zu einem Zeitpunkt *t* befindet sich der Roboter (Agent) und die Welt (Umgebung) in einem Zustand $z_t \epsilon Z$.
Als einen Zustand wird die Gesamtheit des Systems, aus Agent und der Umgebung betrachtet. Da lediglich der Roboterarm (Agent) hier in Bewegung tritt und somit für eine Veränderung im System sorgt, ist der Zustand nur vom Roboterarm abhängig.

Bei der Ausführung einer Aktion $a_t \epsilon A$ zum Zeitpunkt $t$ findet eine Veränderung der Welt statt und somit ein Übergang des Zustandes $z_t$ zum Nachfolgezustand $z_{t+1}$. Der Nachfolgezustand wird durch die Übergangsfunktion ($\delta(z_t, a_t) = z_{t+1}$) definiert, die vom aktuellen Zustand und der ausgeführten Aktion abhängig ist.

Eine direkte Belohnung (engl. immediate reward) $r_t = r(z_t, a_t)$ wird dem Agenten aus der Umgebung zugewiesen. Die direkte Belohnung ist vom aktuellen Zustand und der zuvor ausgeführten Aktion abhängig.
Der Agent erfährt erst nach der ausgeführten Aktion in welchem neuen Zustand ihn diese versetzt, sowie den Nutzen aus dieser Transition und die damit verbundene Belohnung (Abbildung 5).

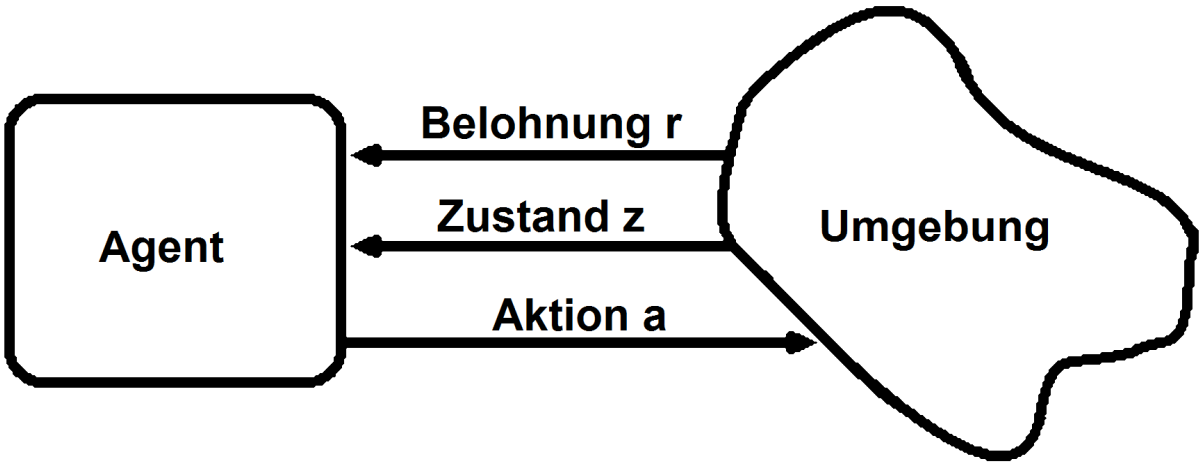



**Abbildung 5: Agent in Wechselwirkung mit der Umgebung [Ert09]**

Zur Erklärung wird die Auflösung des Zustandsraumes, in dem sich der Roboterarm bewegen kann, mit diskreten Werten auf ein 3 x 3-Feld begrenzt (Abbildung 6). Es existieren somit neun Zustandsmöglichkeiten von Agent und Umgebung. Die Bezeichnung der Zustände erfolgt numerisch und kann hier von oben-links nach unten-rechts der Position des Roboterarmes zugeordnet werden, da dieser als einziger für eine Veränderung im System sorgt.

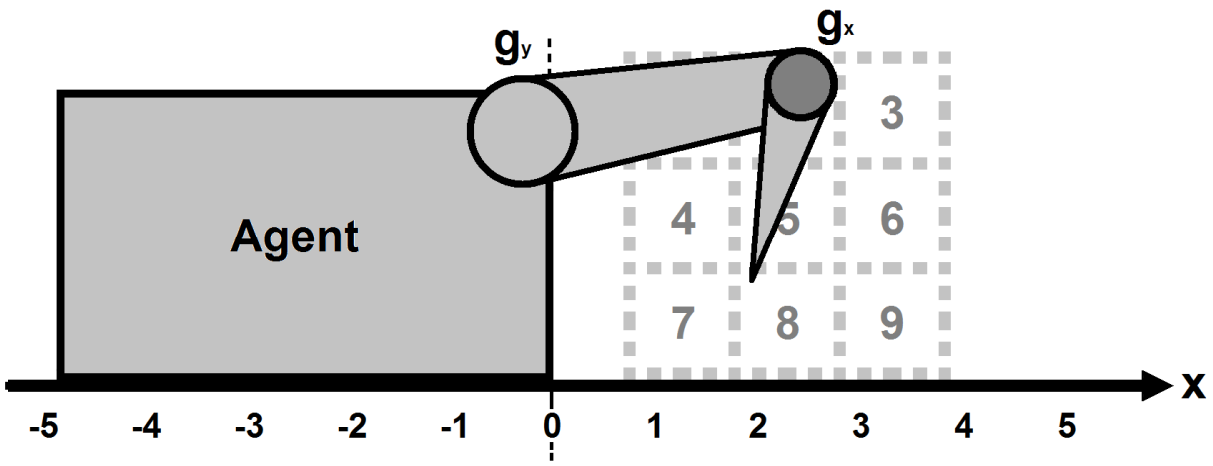



**Abbildung 6: Zustandsraum des Roboterarms**

Ziel für den Roboter (Agent) ist eine Vorwärtsbewegung, die als eine positive Änderung bezeichnet wird und daraufhin belohnt wird. Bewegt sich der Roboter rückwärts, so wird eine negative Änderung vollzogen, auf die eine Bestrafung folgt.
Für eine Zustandsänderung besitzt der Agent die Aktionen *A* aus hoch, runter, vor und zurück, mit dem der Roboterarm aus Sicht des Agenten bewegt werden kann.

Damit der Agent das Ziel der Vorwärtsbewegung ausüben kann, muss dieser eine Strategie $\pi$ entwickeln, die den Roboterarm in einer Abfolge von Zuständen bewegt.

In einem Zustandsraum von 3x3-Feldern befinden sich vier Zustände (1, 3, 7 und 9) mit je zwei Aktionsmöglichkeiten, vier Zustände (2, 4, 6 und 8) mit je drei Aktionsmöglichkeiten und ein Zustand (5) mit allen vier Aktionsmöglichkeiten. Daraus ergeben sich nach der Berechnung $2^4 \times 3^4 \times 4 = 5184$ verschiedene Strategiemöglichkeiten, die der Roboterarm in diesem Zustandsraum ausüben kann.

Die optimale Strategie wird mit $\pi^*$ bezeichnet und wird durch die Maximierung der Belohnung über viele Schritte erreicht. Ziel des Agenten ist es demnach, die Belohnung zu maximieren. Dafür wird die abgeschwächte Belohnung *V* (auch Wert genannt) eingeführt.
Die abgeschwächte Belohnung setzt sich aus der Summe der direkten Belohnung $r_t$ und einer nachfolgenden abgeschwächten Belohnung $r_{t+i}$ zusammen. Um nachfolgende Belohnungen geringer ausfallen zu lassen wird der Diskontierungsfaktor $\gamma$ mit einer Wertigkeit zwischen 0 und 1 eingeführt.

$$V(z_t) = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \ldots = \sum_{i=0}^{\infty} \gamma^i \cdot r_{t+i} \qquad (2.1.1)$$

Die optimale Strategie soll durch den Einsatz der Wert-Iteration oder des Q-Learning ermittelt werden. Für beide Fälle wird eine Vorbereitung benötigt.
Die Vorwärtsbewegung des Roboters kann nur dann ermöglicht werden, wenn der Roboterarm die Übergangsfunktionen *δ(9,West)* und *δ(8,West)* ausübt, also den Übergang von Zustand 9 zu 8 und den Übergang von Zustand 8 zu 7. Diese beiden Übergänge werden mit einer positiven Bewertung ausgelegt.
Genau entgegengesetzt verrichtet der Roboter eine Rückwärtsbewegung aus, wodurch für die Übergangsfunktionen *δ(7,Ost)* und *δ(8,Ost)* eine negative Bewertung eingesetzt wird.

In Abbildung 7 ist der Zustandsraum mit den direkten Belohnungen $r_t = r(z_t,a_t)$ in Abhängigkeit von Zustand und Aktion als graue Pfeile dargestellt. Bei der Ausführung einer Aktion *a* im Zustand *z* wird die darauffolgende Belohnung dem Zustand *z* zur Erinnerung zugeordnet.

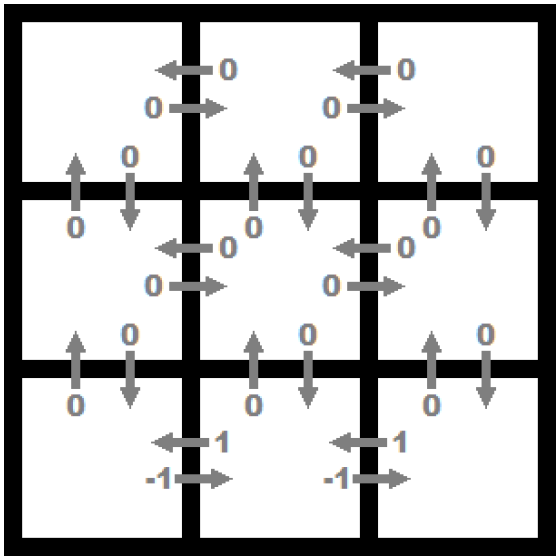



**Abbildung 7: Zustandsraum mit direkter Bewertung der Transitionen [Ert09]**

Für eine optimale Strategie gilt der Ansatz, dass diese Strategie besser oder zumindest genauso gut ist wie jede andere Strategie.

$$V^* = V^{\pi^*}(z) \geq V^{\pi}(z) \qquad (2.1.2)$$

### 2.1.1 Wert-Iteration

Um eine optimale Strategie zu erhalten, muss die Belohnung durch den Agenten maximiert werden. Das Bellman-Prinzip sagt aus: „Unabhängig vom Startzustand $s_t$ und der ersten Aktion $a_t$ müssen ausgehend von jedem möglichen Nachfolgezustand $s_{t+1}$ alle folgenden Entscheidungen optimal sein.“[1] (s=z).
Somit muss jede Teilbelohnung innerhalb von $\pi^*$ ebenfalls die maximale Belohnung enthalten. Nach den Gleichungen 2.1.1 und 2.1.2 ergibt sich damit folgender Ansatz.

$$V^*(z_t) = \max_{a_t, a_{t+1}, a_{t+2}, \dots} (r(z_t, a_t) + \gamma \cdot r(z_{t+1}, a_{t+1}) + \gamma^2 \cdot r(z_{t+2}, a_{t+2}) + \dots) \tag{2.1.3}$$

Da die direkte Belohnung vom aktuellen Zustand und der aktuellen Aktion abhängig ist, kann diese von der abgeschwächten Belohnung, die von den Nachfolgezuständen und Nachfolgeaktionen abhängig ist, ausgeklammert werden.

$$V^*(z_t) = \max_{a_t} [r(z_t, a_t) + \gamma \cdot \max_{a_{t+1}, a_{t+2}, \dots} (r(z_{t+1}, a_{t+1}) + \gamma^2 \cdot r(z_{t+2}, a_{t+2}) + \dots)] \tag{2.1.4}$$

Ersichtlich wird nun die rekursive Charakteristik aus dieser Gleichung. Der Diskontierungsfaktor wurde ebenfalls aus der abgeschwächten Folgebelohnung ausgeklammert, wodurch folgende Rekursive Gleichung entsteht.

$$V^*(z_t) = \max_{a_t} [r(z_t, a_t) + \gamma \cdot V^*(z_{t+1})] \tag{2.1.5}$$

Da der Nachfolgezustand mit der Übergangsfunktion δ beschrieben werden kann, ergibt sich eine rekursive Gleichung die ausschließlich vom aktuellen Zustand sowie der aktuell gewählten Aktion abhängig ist.

$$V^*(z) = \max_{a} [r(z, a) + \gamma \cdot V^*(\delta(z, a))] \tag{2.1.6}$$

[1] Richard Bellman 1957, zit. nach Wolfgang Ertel, Grundkurs Künstliche Intelligenz 2009, S. 290

Diese Gleichung wird als Bellman-Gleichung bezeichnet, da sie dem Bellman-Prinzip entspricht. Die maximale Belohnung ergibt sich durch die abgeschwächte Folgebelohnung und lokal durch die optimale Aktion *a* im Zustand *z*.

Für die optimale Strategie $\pi^*$ wird im Zustand *z* die Aktion mit der maximalen Belohnung $V^*$ ausgewählt.

$$\pi^*(z) = \arg\max_a [r(z,a) + \gamma \cdot V^*(\delta(z,a))] \tag{2.1.7}$$

Zur Berechnung der maximalen Belohnung, nach der sich die optimale Strategie orientiert, wird eine gleichmäßige Aktualisierung aller Zustände nach der Vorschrift

$$V(z) = \max_a [r(z,a) + \gamma \cdot V(\delta(z,a))] \tag{2.1.8}$$

benötigt, bis $V^*$ für jeden Zustand z erreicht ist. Da jeder Zustand auch einen Nachfolgezustand darstellen kann, muss für den Wert $V(z)$ ein Startwert initialisiert werden.
Es folgt ein rekursiver Zugriff auf jeden Zustand, bei dem die Gleichung 2.1.8 angewendet wird und für den Wert $V(\delta(z,a))$ der beste Nachfolgezustand gewählt wird. Dieses Verfahren wird Wert-Iteration genannt, bei dem der Wert $V^*$ konvergiert.

*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 Wert-Iteration [Ert09]**

Auf das Beispiel des Laufroboters bezogen ergeben sich folgende Strategien, nachdem der Wert konvergiert. Im linken Zustandsraum ist die maximale Belohnung $V^*$

jedem Zustand zugeordnet und die direkte Belohnung mit grauen Pfeilen und der dazugehörigen Wertigkeit dargestellt. Rechts befinden sich zwei optimale Strategien, die sich aus der Gleichung 2.1.7 ergeben.

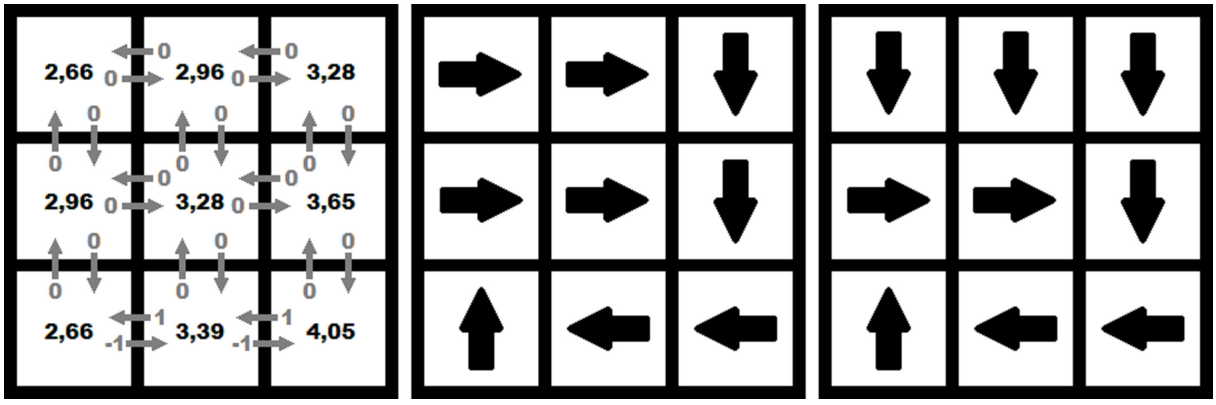



**Abbildung 9: Wert-Iteration angewendet an einen 3x3 Zustandsraum, links die Zuordnung von V* zu den Zuständen, rechts zwei optimale Strategien [Ert09]**

Zu beachten ist, dass die optimale Strategie eine Addition aus *V** und der direkten Belohnung ist. Für die Anwendung der optimalen Strategie müssen dem Agenten die direkten Belohnungen und die Übergangsfunktionen bekannt sein.

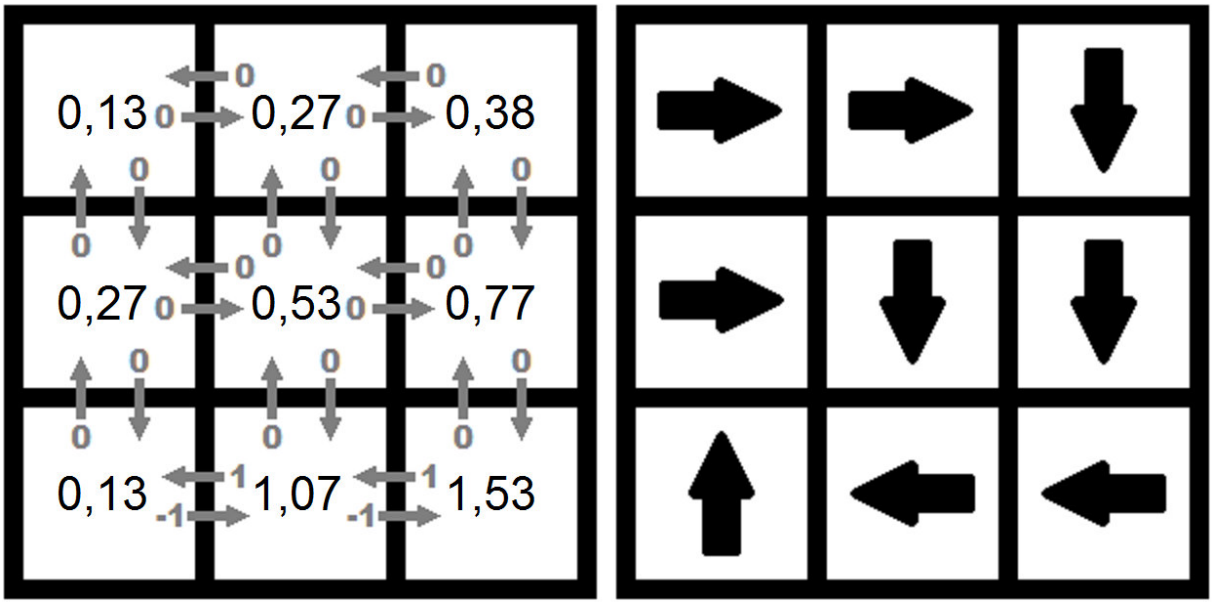



**Abbildung 10: Wert-Iteration mit einem Diskontierungsfaktor von 0,5**

Für die oben angegebenen Werte und der daraus folgenden Strategie beträgt der Diskontierungsfaktor $\gamma = 0{,}9$. Bei $\gamma = 0{,}5$ wird eine kürzere Strategie erreicht (Abbildung 10).

Ein kleinerer Diskontierungsfaktor führt zu einer schnelleren Konvergenz und damit Beendigung des Algorithmus, wobei nicht die optimale Strategie erreicht wird. Für eine optimale Strategie wird demnach ein höherer Zeitaufwand benötigt.

### 2.1.2 Q-Lernen

Damit das erlernte Wissen auch ohne Model der direkten und abgeschwächten Belohnung angewendet werden kann, wird die Bewertungsfunktion $Q(z_t, a_t)$ eingeführt. Durch die Bewertungsfunktion $Q(z_t, a_t)$ findet eine Bewertung der Aktion im jeweiligen Zustand statt. Die Strategie wird nach folgender Gleichung ausgelegt.

$$\pi = \arg\max_{a} Q(z,a) \tag{2.1.9}$$

Wie in der Wert-Iteration wird das Schema der abgeschwächten Belohnung beibehalten. Da die resultierende Belohnung auf Zustand und Aktion abgebildet wird, fällt das Maximum der direkten Aktion weg. Ausgehend von Gleichung 2.1.1 entsteht folgender Ansatz.

$$Q(z_t,a_t) = \max_{a_{t+1},a_{t+2},\dots} (r(z_t,a_t) + \gamma \cdot r(z_{t+1},a_{t+1}) + \gamma^2 \cdot r(z_{t+2},a_{t+2}) + \dots) \tag{2.1.10}$$

Nachfolgend werden die direkte Belohnung und der Diskontierungsfaktor ausgeklammert. Es folgt erneut eine rekursive Gleichung.

$$Q(z_t,a_t) = r(z_t,a_t) + \gamma \cdot \max_{a_{t+1},a_{t+2},\dots} (r(z_{t+1},a_{t+1}) + \gamma \cdot r(z_{t+2},a_{t+2}) + \dots)$$

$$Q(z_t,a_t) = r(z_t,a_t) + \gamma \cdot \max_{a_{t+1}} (r(z_{t+1},a_{t+1}) + \gamma \cdot \max_{a_{t+2}} (r(z_{t+2},a_{t+2}) + \dots))$$

$$Q(z_t,a_t) = r(z_t,a_t) + \gamma \cdot \max_{a_{t+1}} Q(z_{t+1},a_{t+1})$$

$$Q(z_t,a_t) = r(z_t,a_t) + \gamma \cdot \max_{a'} Q(\delta(z,a),a') \tag{2.1.11}$$

Gleichgesetzt zur Wert-Iteration ist die direkte Belohnung abhängig vom aktuellen Zustand und der gewählten Aktion. Die nachfolgende abgeschwächte Belohnung ist abhängig von der Übergangsfunktion und dem damit verbundenen Nachfolgezustand sowie dem Maximum aus den Aktionen des Nachfolgezustandes.

Wie bereits bei der Wert-Iteration wird diese Gleichung ebenfalls in einer Iterationsschleife angewendet. Die Q-Matrix, die am Ende des Iterationsverfahrens die Lösung darstellt, wird zu Beginn mit zufälligen Werten oder mit Null initialisiert.
In der ersten Schleife wird ein Zufallszustand z gewählt, von dem aus mit der Gleichung 2.1.11 begonnen wird. In der Inneren Schleife wird die für die Gleichung benötigte Aktion *a* ermittelt. Nach der Anwendung der Gleichung wird der Nachfolgezustand als der aktuelle Zustand gesetzt und erneut wird die innere Schleife durchlaufen, bis ein Endzustand oder ein Zeitlimit erreicht wird.

Der Durchlauf der inneren Schleife bis zum Abbruchkriterium wird als eine Episode bezeichnet. Eine Episode enthält eine Zustandsfolge, die der Agent mit der Umgebung nacheinander einnehmen kann.

*Q-Lernen()*
*Für alle z ϵ Z, a ϵ A*
$Q(z,a) = 0$ *oder zufällig*
*Wiederhole*
*Wahl eines Zustandes z (zufällig)*
*Wiederhole*
*Wahl und Ausführung einer Aktion a*
*Erhalte Belohnung r und neuen Zustand z'*
$Q(z_t, a_t) = r(z_t, a_t) + \gamma \cdot \max_{a'} Q(\delta(z,a), a')$
$z := z'$
*Abbruch z ist ein Endzustand oder Zeitschranke erreicht*
*Abbruch Q konvergiert*

**Abbildung 11: Algorithmus für das Q-Lernen [Ert09]**

Die Abbruchbedingung der inneren Schleife ist hier auf die Zeitschranke beschränkt, da ein Endzustand nicht erreicht werden kann. Der Laufroboter hat die Aufgabe sich stetig fort zu bewegen und nicht nur einen Satz (Schritt) zu bewältigen.
Das Zeitlimit wird über die Anzahl der aufeinanderfolgenden Schritte definiert. Ist eine gewisse Anzahl an Schritten erreicht, so ist die Abbruchbedingung erfüllt.

Das Kriterium für die Auswahl der Aktion a ist nicht im Algorithmus angegeben. Zu Beginn erfolgt eine zufällige Wahl der Aktion. Das bedeutet, der Nachfolgezustand und die damit verbundene Verteilung der Belohnung erfolgt rein zufällig. Bei unendlichem Durchlauf des Algorithmus wird jede Transition gleichermaßen durchlaufen und eine gleichmäßige Verteilung der Belohnung ermöglicht.
In Folge dessen, wird die Auffindung der optimalen Strategie aus dem gesamten Zustandsraum ermöglicht. Der Nachteil ist die Zeit, die benötigt wird bis die Q-Matrix konvergiert und damit zu einem Abbruch des Algorithmus führt, da auch für die optimale Strategie weniger interessante Zustände mehrmals durchlaufen werden.
Die Anwendung der rein zufälligen Wahl der Aktion a wird als Lernen bezeichnet.

Um eine schnellere Konvergenz umzusetzen, wird neben dem Lernen zunehmend auch das Verwerten eingesetzt. Dabei wird die Aktion zunehmend in Abhängigkeit von bereits gelerntem Wissen gewählt.
Die Q-Matrix, die im Laufe zunehmender Iterationsschritte mit Werten gefüllt wird, kann frühzeitig eine Strategie enthalten. Zur Festigung dieser Strategie wird als Aktion das Argument des Maximums der möglichen Aktionen gewählt. Der Vorteil ist eine schnelle Konvergenz und damit Beendigung des Algorithmus. Nachteil ist, dass eventuell nicht jede Strategie, eingeschlossen die optimale Strategie, aufgedeckt wird.

Sowohl das Lernen, wie auch das Verwerten haben ihre Vor- und auch Nachteile. Bei der Anwendung des Algorithmus des Q-Lernen wird zu Beginn das Lernen und zunehmend mit der Iterationsanzahl ein Übergang zum Verwerten eingesetzt.

In der Praxis hat der Laufroboter bewiesen, dass sogar unter Beeinträchtigung der Fortbewegungsmittel dennoch die Umsetzung der Fortbewegung, unter Anwendung

des Markov-Entscheidungsprozesses, realisiert werden konnte. Mit einem defekten Servo entsprach dies nicht der eigentlich zu erwartenden optimalen Bewegung, dennoch wurde die Fortbewegung (als humpeln) realisiert.

## 2.2 Beispiel 2: Forstwirtschaft

In diesem Beispiel wird die Forstwirtschaft unter der Betrachtung der Erhaltung des ökologischen Naturhaushalts und dem wirtschaftlichen Verkauf von Holz behandelt.
Es existieren dafür zwei Aktionsmöglichkeiten, zum einen das Warten, damit die Ressource an Holz wächst und als zweites das Abholzen, um die Ressource wirtschaftlich zu nutzen.
Es werden drei Zustände definiert. Jeder Zustand betrachtet eine Zeitspanne der wachsenden Bäume von 20 Jahre. Der erste Zustand enthält das Alter von 0-20 Jahre, der zweite Zustand von 21-40 Jahre und der dritte Zustand jedes Alter von über 40 Jahren.

### 2.2.1 Temporal-Difference-Learning

Äußere Einflüsse wurden im bisherigen Beispiel nicht mit einbezogen. In der Praxis ist die Übergangsfunktion nicht konstant stabil und kann von zufälligen Ereignissen aus der Umwelt beeinflusst werden.
Damit das Lernen in nichtdeterministischer Umgebung ermöglicht werden kann, wird im zweiten Beispiel die Erweiterung der Gleichungen für das Temporal-Difference-Learning betrachtet.

Der Ausbruch eines Waldbrandes stellt eine ungewollte Abweichung dar. Es wird angenommen, dass dieser Fall mit 10% Wahrscheinlichkeit innerhalb eines Zeitraumes eintritt. Am Ende eines Zustandes besteht bei dem Übergang zum nächsten Zustand eine ungewollte Abweichung, die den abgebrannten Wald in den ersten Zustand zurückversetzt. Abbildung 12 soll das Schema der Zustände und Transitionen mit deren direkten Belohnung und Wahrscheinlichkeit darstellen.

Im Beispiel des Laufroboters handelte es sich um eine deterministische Umgebung, in der jede ausgeführte Aktion zu 100% in den entsprechenden Nachfolgezustand geführt hat. Im Beispiel der Forstwirtschaft wird eine nichtdeterministische Umgebung betrachtet, daher sind die Wahrscheinlichkeiten bei der Aktion Warten, die zu dem entsprechenden Nachfolgezustand führen, in Klammern angegeben. Bei der Aktion Roden beträgt diese Wahrscheinlichkeit 100% und wird daher nicht weiter angegeben.

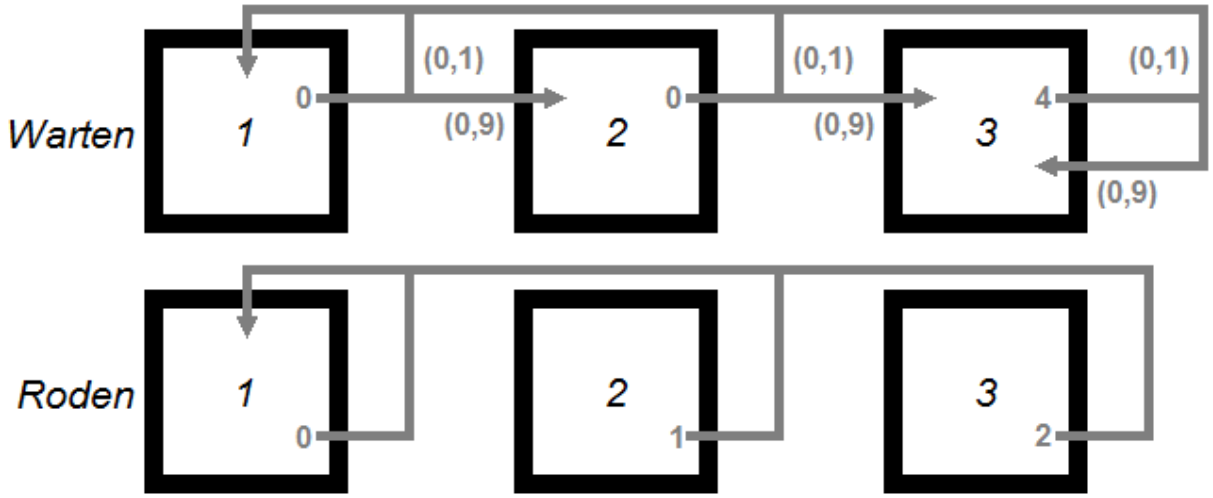



**Abbildung 12: Zustände und Transitionen der Forstwirtschaft**

Die direkte Belohnung für die ausgeführte Aktion des Wartens in den Zuständen 1 und 2 wurden auf 0 gesetzt. Durch die Aktion des Wartens besteht das Ziel darin, über einen langfristigen Zeitraum einen ökologischen Haushalt zu bieten und den maximalen Erfolg für die Umsetzung der zweiten Aktion zu erzielen. Für den letzten und hierbei dritten Zustand wird daher eine Belohnung von 4 verteilt. Es handelt sich dabei um einen Endzustand des Wartens.

Bei der Betrachtung, dass das Roden neu gepflanzter Bäume ineffizient sei, entfällt die direkte Belohnung dieser Aktion im ersten Zustand. In dem darauffolgenden Zustand wird eine Belohnung von 1 und im letzten Zustand, bei der die Aktion des Rodens am effizientesten ist, mit einer Wertigkeit von 2 verteilt.

Unter der Anwendung der bisherigen Gleichungen ist durch die nicht stabile Übergangsfunktion keine Konvergenz gesichert. Bei konstanter Wahl von Zustand und Aktion entstehen unterschiedliche Nachfolgezustände. Da der Nachfolgezustand

entscheidend für die abgeschwächte Belohnung ist, muss zu den alternierenden Werten ein Ausgleich geschaffen werden.
Dieser Ausgleich wird durch die Erweiterung der Gleichung 2.1.11 mit zwei Bestandteilen ermöglicht. Zum einen die Addition der vorherigen Belohnung des Ausgangszustandes und einem variablen Gewichtungsfaktor *α*, der von den Iterationsschritten abhängig ist. Dieser Gewichtungsfaktor soll abnehmend auf den bisherigen Bestandteil der Gleichung 2.2.1 (Teil in eckiger Klammer) und zunehmend auf den hinzugefügten Bestandteil, der Addition der Belohnung des Ausgangszustandes wirken.

$$Q_n(z,a) = (1-\alpha) \cdot Q_{n-1}(z,a) + \alpha_n \cdot [r(z,a) + \gamma \cdot \max_{a'} Q(\delta(z,a),a')] \qquad (2.2.1)$$

Ohne Gewichtungsfaktor *α* wäre eine fortlaufende Steigerung der Belohnung die Folge und somit eine Konvergenz ausgeschlossen. Der Gewichtungsfaktor muss mit zunehmenden Iterationsschritten abnehmen. Um eine gleichwertige Behandlung aller Übergangsfunktionen zu gewährleisten, müssen die Iterationsschritte des Gewichtungsfaktors *α* von gewähltem Zustand und Aktion abhängig sein.

$$\alpha_n = \frac{1}{1 + b_n(z,a)} \qquad (2.2.2)$$

Durch Umstellung der Gleichung 2.2.1 nach *α* ist zu erkennen, dass der vom Gewichtungsfaktor abhängige Teil die Differenz der Belohnung zum vorherigen Q-Wert darstellt.

$$Q_n(z,a) = Q_{n-1}(z,a) + \alpha_n \cdot [r(z,a) + \gamma \cdot \max_{a'} Q(\delta(z,a),a') - Q_{n-1}(z,a)] \qquad (2.2.3)$$

Für *α=1* entsteht die Ausgangsgleichung 2.1.11, wodurch die Verwendung dieser Gleichung unter nichtdeterministischer Umgebung entfällt. Ist *α=0,* so entfällt die Belohnungsdifferenz und es käme zu keinen Wertveränderungen. Demnach muss der Gewichtungsfaktor einen Wert zwischen null und eins betragen.

Das Ergebnis des Beispiels der Forstwirtschaft, unter Betrachtung der oben angegebenen Belohnungsverteilung, ergibt für das Iterations-Verfahren, wie für das Q-Lernen eine Lösung, die besagt, dass in allen Zuständen die Aktion Warten gewählt werden soll. Die Aktion Roden entfällt, was dazu führen würde, dass eine wirtschaftliche Nutzung des Waldes nicht möglich wäre.
Die Aktion Warten verteilt im dritten Zustand eine höhere direkte Belohnung, als die Aktion Roden, wodurch diese Aktion auch mit 10% Wahrscheinlichkeit einer Abweichung gewählt wird.
Bei Änderung der direkten Belohnung von 4 auf 0,4 für die Aktion Warten des dritten Zustandes, entsteht eine Lösungsstrategie, bei der für die ersten beiden Zustände das Warten und im dritten Zustand die Aktion Roden gewählt werden soll.

Tritt der Fall ein, dass die Waldbrandgefahr auf eine Wahrscheinlichkeit von 80% ansteigt, so errechnet sich eine Lösungsstrategie, bei der im ersten Zustand die Aktion Warten und bereits im zweiten Zustand die Aktion Roden gewählt werden soll.

# 3 Referenzapplikation

## 3.1 Nachbildung von Theseus

Anhand des Beispiels des Laufroboters (Kapitel 2.1) kann ein Bezug zur eigentlichen Aufgabenstellung gezogen werden. Bei der Betrachtung des Zustandsraums ist zu erkennen, dass der Agent mit der Bewegung des Roboterarmes Ähnlichkeiten zur Bewegung eines Objektes im Labyrinth aufweist.

Das Labyrinth besitzt eine kartesische Anordnung, sowie die Ausführung der Aktionen des Roboterarmes. Die Spitze des Roboterarmes, die einen bestimmten Zustand einnehmen kann, wird durch die Maus ersetzt. Statt einer Betrachtung von der Seite, kann der Zustandsraum aus der Vogelperspektive betrachtet werden.
Ein Zustand wird durch Agent und Umwelt beschrieben. Da lediglich die Bewegung der Maus für eine Veränderung sorgt, existieren so viele Zustände, wie die Maus an Positionen einnehmen kann. Somit kann jeder Zustand grafisch einer festen Position der Maus zugeordnet werden.

Die Grenze des Zustandsraumes ist bei dem Laufroboter auf die Reichweite des Armes und der Bewegungsweite der Gelenke beschränkt. Die Maus ist im Bezug zum Zustandsraum nicht durch ihre eigenen motorischen Fähigkeiten eingeschränkt.
Der Zustandsraum im Labyrinth ist somit durch ein Rahmen aus Hindernissen eingegrenzt.

Die vier Aktionsmöglichkeiten sollen zu Beginn aus den Bewegungsrichtungen Nord, Ost, Süd und West bestehen. Das Ziel der Maus ist ein Punkt im Labyrinth zu finden. Im Gegenzug zum Laufroboter, existiert hier ein Endzustand, da die Maus nach dem Erreichen des Zieles keine weiteren Aktionen ausführt.

## 3.2 Simulationsmodell

Für die rechnerische Umsetzung wird das Rechenprogramm Matlab mit einer bereits vorhandenen MDP-Toolbox verwendet. In der Toolbox ist bereits der Algorithmus der Wert-Iteration und Q-Lernen implementiert [Cha14].

Die Struktur des zu bearbeitenden Labyrinths wird mit Zahlen in Matrizenform dargestellt und über das Paint-Programm im 4Bit-Format eingelesen. Insgesamt vier verschiedene Farbwerte (bzw. Zahlenwerte in Matlab) beschreiben den Aufbau des Labyrinths. Die im Paint dargestellten Farbwerte (aus der Standardfarbpalette) geben an, welche Eigenschaft jedes Feld besitzt. Sie können aus einer frei begehbaren Fläche (weiß / 15), einem Hindernis (braun / 7), dem Startpunkt der Maus (rot / 9) und dem Zielpunkt (grün / 6) bestehen. Ein Pixel entspricht einem Feld des Labyrinths.

Folgendes (stark vereinfachtes) Labyrinth, mit einer Größe von 2 x 4 Feldern, soll für die nächsten Erklärungen verwendet werden. Der Rand besteht aus einem Hindernisrahmen, der zur Abgrenzung des Labyrinths dient. Die Zustände werden von oben links nach unten rechts durchnummeriert. Hindernisse werden dabei übersprungen, da vorerst eine statische Betrachtung des Labyrinths angenommen wird. Ein Zustand, in dem die Maus die Position eines Hindernisses einnehmen kann, existiert nicht. Es wird ein Zustandsraum mit 7 Zuständen erfasst.

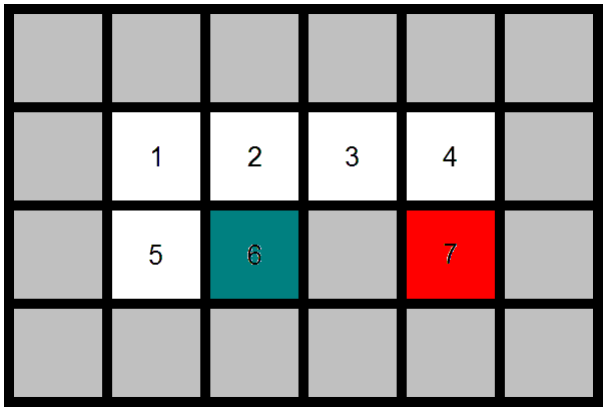



**Abbildung 13: Labyrinthstruktur mit 7 Zuständen, rot=Start, grün=Ziel**

### 3.2.1 Modellierung der Wissensquellen

Als Vorbereitung müssen zur Anwendung der Algorithmen zwei Matrizen erstellt werden. Die erste Matrix wird als Belohnungsmatrix (Reward-Matrix/R-Matrix) bezeichnet. Sie stellt eine Tabelle dar, die abgerufen wird, um den aktuellen, direkten Belohnungswert zu erhalten. Da die Belohnung von Zustand und Aktion abhängig ist, orientiert sich der Zeilenvektor nach dem Zustand und der Spaltenvektor nach der Aktion. Die direkte Belohnung wird nur bei der Transition in den Zielzustand vergeben, somit muss dieser Wert auf den vorherigen Zustand mit der entsprechenden Aktion, die in den Zielzustand führt, gesetzt werden.

| | | Aktion a | | | |
|---|---|---|---|---|---|
| | | Nord | Ost | Süd | West |
| Zustand z | 1 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 1 | 0 |
| | 3 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 | 0 |
| | 5 | 0 | 1 | 0 | 0 |
| | 6 | 0 | 0 | 0 | 0 |
| | 7 | 0 | 0 | 0 | 0 |

**Tabelle 1: R-Matrix mit Belohnungswert**

Diese Bedingung wird im Beispiel von den zwei Übergangsfunktionen *δ(2, Süd)* und *δ(5, Ost)* erfüllt. Beide Übergangsfunktionen erhalten eine direkte Belohnung von 1.

Die R-Matrix ist statisch und somit ein Teil der Umwelt. Innerhalb der R-Matrix kommt es über den gesamten Algorithmus zu keinen Veränderungen.
Zur Speicherung der veränderten Belohnungswerte wird in der Wert-Iteration ein Wert-Vektor (V) mit der Größe des Zustandsraumes und bei dem Q-Lernen eine Q-Matrix, mit der gleichen Größe der R-Matrix, verwendet. Die Q-Matrix ist dynamisch und enthält nach dem Durchlauf des Q-Lernen-Algorithmus die Lösungsstrategie.

Unter der Bedingung, dass keine Fallen oder andere negativen Aspekte vorhanden sind, ist die Maus keiner Gefahr im Labyrinth ausgesetzt. Demnach existieren keine negativen Belohnungen in diesem System. Die einzige Belohnung wird bei der Transition in den Nachfolgezustand, in dem die Maus das Ziel erreicht, vergeben.

Die zweite Matrix stellt die probabilistischen Übergangsfunktionen dar und wird daher als Übergangsmatrix (P-Matrix) bezeichnet. Mit ihr werden die Möglichkeiten der Übergänge von einem Zustand in den Nachfolgezustand dargestellt und somit die Zusammenhänge der einzelnen Zustände zueinander. Der Zeilenvektor orientiert sich ebenfalls nach dem aktuellen Zustand und besitzt somit die gleiche Länge wie die R-Matrix. Der Spaltenvektor stellt den Nachfolgezustand dar. Da jeder Zustand einen möglichen Nachfolgezustand darstellen könnte, ist der Spaltenvektor genauso lang wie der Zeilenvektor.
Eine P-Matrix definiert die Zusammenhänge der Zustände und deren Nachfolge-zustände für eine Aktion. Für vier Aktionen bestehen demnach vier P-Matrizen.

Für das Labyrinth aus Abbildung 13 existiert für jede Aktion eine 7 x 7-P-Matrix. Nachfolgend soll das Prinzip der P-Matrix lediglich an der Aktion *a=Nord* erklärt werden. Bei einer horizontalen und vertikalen Bewegungsmöglichkeit besitzt jeder Zustand maximal vier angrenzende Zustände, die einen möglichen Nachfolge-zustand darstellen können.

| Aktion a=Nord | | Nachfolgezustand z‘ | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Zustand z | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | 5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 6 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 7 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

**Tabelle 2: P-Matrix der Aktion Nord**

Ausgehend von Zustand *z=5* ist bei der Aktion *a=Nord* der Nachfolgezustand *z‘=1*. Die P-Matrix besitzt in der fünften Zeile, die dem aktuellen Zustand entspricht, eine eins in der ersten Spalte, die dem Nachfolgezustand entspricht. Die restlichen Werte der fünften Zeile sind null. Für die gleiche Aktion und den aktuellen Zustand *z=1* ergibt sich durch das im Norden befindliche Hindernis der Nachfolgezustand *z‘=1*. Der aktuelle Zustand ist zugleich der Nachfolgezustand, wodurch in der ersten Zeile eine Eins in der ersten Spalte gesetzt wird.

Zu erkennen ist auch, dass die Spalten 5, 6 und 7 komplett null sind. Es existiert keine Möglichkeit mit der Aktion Nord in diese Zustände zu gelangen. Die Darstellung der gesamten P-Matrix über alle Aktionen befindet sich im Anhang (Siehe Anhang: Tabelle 4).

Bei einer Wahrscheinlichkeitsbetrachtung kann der Wert prozentual auf die entsprechenden möglichen Nachfolgezustände in der Zeile aufgeteilt werden (Siehe Kapitel 3.2.4 Temporal-Difference-Learning). Als ein Kriterium muss die Summe jeder Zeile eins ergeben, da die Summe der Wahrscheinlichkeiten aller möglichen Nachfolgezustände 100% nicht überschreiten darf.

Für jede Aktion wird zur Erstellung der P-Matrix eine zeilen- und spaltenweise Abfrage der Felder mit deren angrenzenden Nachbarfelder vorgenommen. Das Ergebnis ist eine dreidimensionale Matrix mit dem Format P(z, z‘, a).

### 3.2.2 Wert-Iteration

Zur Darstellung der Wert-Iteration wird das folgende Labyrinth verwendet. In der nachfolgenden Abbildung ist bereits der Lösungsweg mit Start (rot) und dem Ziel (grün) dargestellt.

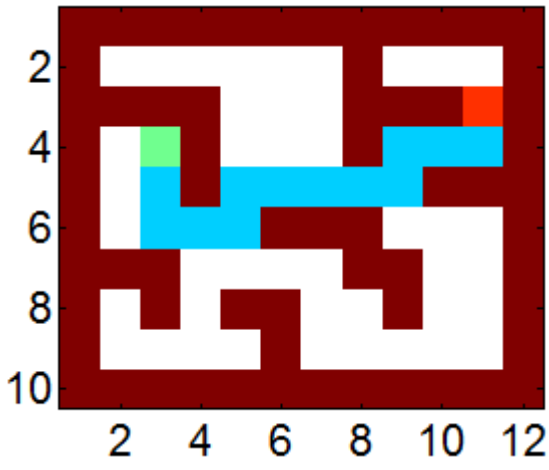



**Abbildung 14: Labyrinth mit Start, Ziel und Lösungsweg**

Die Abbildung 15 stellt den Verlauf der Wert-Iteration dar. Auf der linken Seite ist die maximale Belohnungsdifferenz der einzelnen Zustände in einem Durchlauf des gesamten Zustandsraums nach folgender Formel dargestellt.

$$dV = \max(V_t - V_{t-1}) - \min(V_t - V_{t-1}) \tag{3.2.1}$$

Nach dem ersten Durchlauf wird lediglich die direkte Belohnung mit dem Wert 1 vom Ziel verteilt. Auf darauf folgende Durchläufe kommt der Diskontierungsfaktor zum Einsatz. Die Belohnungsdifferenz fällt kontinuierlich ab und konvergiert gegen null.

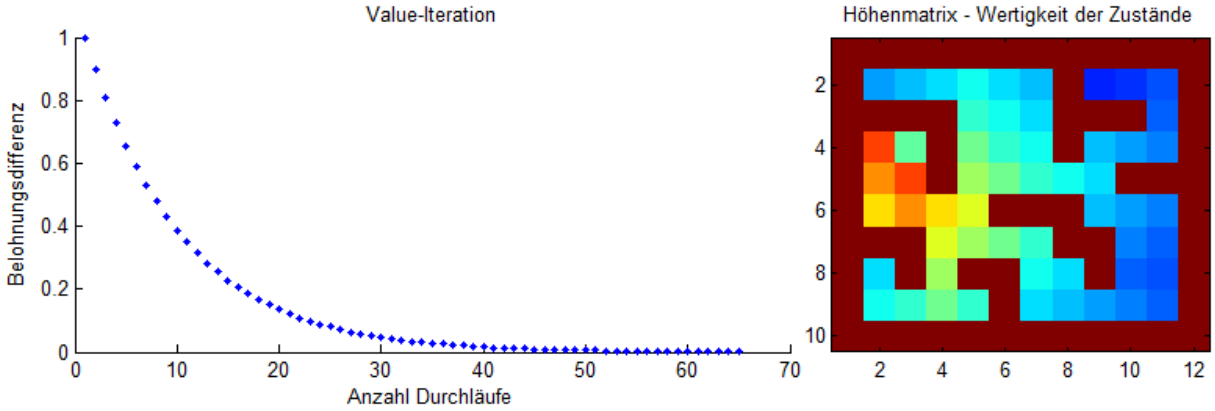



**Abbildung 15: Belohnungsdifferenz und Wertegebirge der Wert-Iteration**

Auf der rechten Seite ist das Wertegebirge der gefüllten Belohnungsmatrix (nach dem Algorithmus) dargestellt. Direkt um das Ziel befindet sich, dargestellt mit einem warmen rötlichen Farbton, die größte Belohnungswertigkeit und geht in den kalten blauen Farbton, mit niedriger Belohnungswertigkeit, über. Bei der Wert-Iteration entsteht ausgehend vom Zielzustand eine gleichmäßige Verteilung der Belohnungswerte.

Der Diskontierungsfaktor ist der einzige Parameter, mit dem eine Änderung am Algorithmus vorgenommen werden kann. Es ist anzunehmen, dass mit kleinerem Diskontierungsfaktor der Algorithmus frühzeitiger konvergiert. Im Diagramm 1 werden drei Kurvenverläufe mit unterschiedlichen Diskontierungsfaktoren dargestellt.

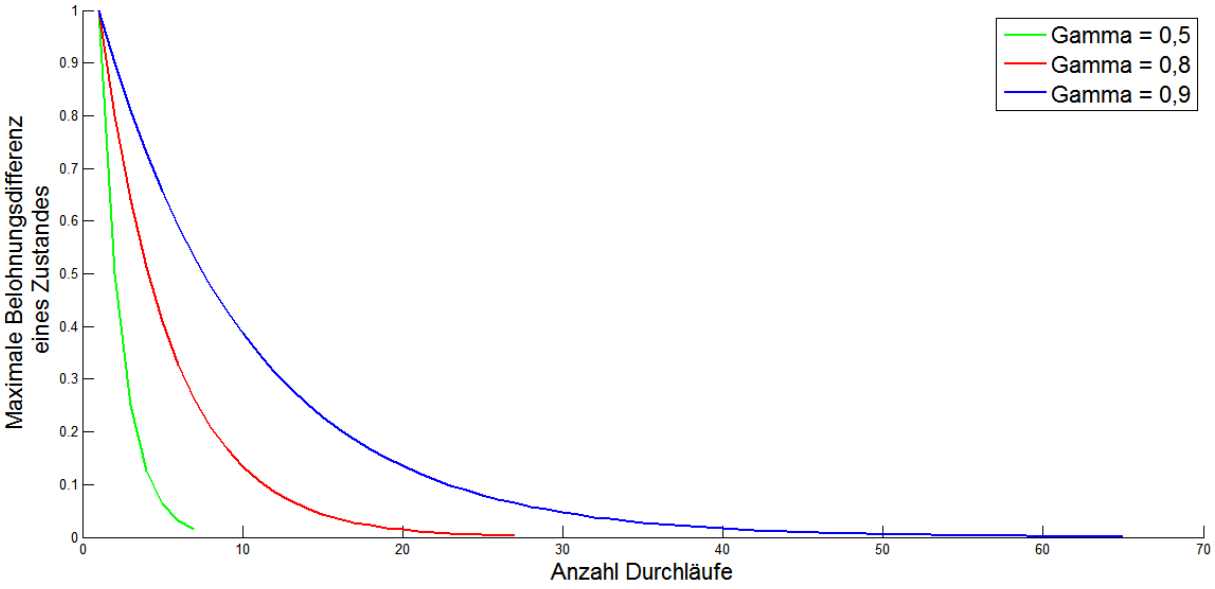



**Diagramm 1: Wert-Iteration mit unterschiedlichen Diskontierungsfaktoren**

Aus dem Diagramm ist zu erkennen, dass ein kleinerer Diskontierungsfaktor die Belohnungsverteilung frühzeitiger konvergieren lässt. Dabei wird jedoch nicht immer das Ziel erreicht. Bei einem Diskontierungsfaktor von 0,5 findet eine zu geringe Verteilung der Belohnungswerte im Zustandsraum statt (Abbildung 16). Es kommt zu früh zu einem Abbruch.

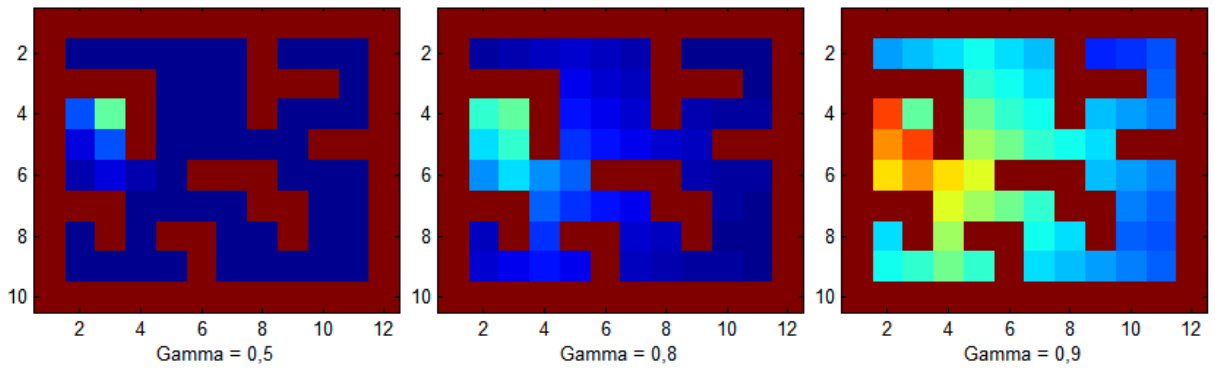



**Abbildung 16: Belohnungsverteilung bei unterschiedlichen Diskontierungsfaktoren**

In allen nachfolgenden Versuchen wird der Diskontierungsfaktor auf 0,9 gesetzt. Somit wird eine ausführliche Verteilung der Belohnungswerte ermöglicht.

Der Vorteil der Wert-Iteration liegt darin, dass die optimale Strategie immer aufgedeckt wird, sofern ein hoher Diskontierungsfaktor gesetzt wurde. Nachteil ist, dass die direkte Belohnung (R-Matrix) zum Ablaufe des Lösungsweges mit benötigt wird. (siehe Gleichung 2.1.7)

### 3.2.3 Q-Lernen

Das Q-Lernen besitzt mehrere Parameter, die zur Einstellung notwendig sind und Einfluss auf die Lösung nehmen. Nachfolgend sollen diese Parameter dargestellt werden.

#### *3.2.3.1 Anzahl der Gesamtiterationen N*

Als Abbruchbedingung des Q-Lernens wurde in Abbildung 11 die Konvergenz der Q-Matrix genannt. Da im Gegenzug zur Wert-Iteration der Nachfolgezustand von der zufälligen Wahl der Aktion abhängig ist, erfolgt eine unregelmäßige Zustandseinnahme.
Ein Konvergenzvergleich nach mehreren Iterationsschritten könnte einen frühzeitigen Abbruch bewirken, da nicht garantiert werden kann, dass Belohnungswerte verteilt wurden. Die Konvergenzbedingung wäre erfüllt und der Algorithmus wäre beendet, bevor dieser überhaupt begonnen hätte.

Zur Realisierung der Konvergenz als Abbruchkriterium müsste vorher sichergestellt werden, dass vor einem Konvergenzvergleich jeder Zustand mindestens einmal besucht wurde. Mit steigender Größe des Zustandsraumes dauert es länger, bis alle Zustände einmal besucht wurden. Um einen Vergleich zu ermöglichen, muss zudem gewährleistet werden, dass bei jedem Konvergenzvergleich die Anzahl der ausgeführten Transitionen konstant sind. Dies ist zu berücksichtigen, da mit steigenden Transitionen angenommen werden kann, dass auch mehr Belohnungswerte verteilt werden.
Bei der Programmierung wird daher eine feste und ausführliche Anzahl an Iterationen als Abbruchbedingung vorgegeben. Durch die Größe des Zustandsraumes und der Einstellung weiterer Parameter muss die Anzahl der Gesamtiterationen angepasst werden.

#### *3.2.3.2 Entscheidungsfunktion zwischen Lernen und Verwerten*

Im Kapitel 2.1.2 Q-Lernen wurde bereits erklärt, dass es ein Wechselspiel zwischen der rein zufälligen Wahl einer Aktion (Lernen) und der Orientierung nach der Aktion mit der maximalen Belohnung (Verwerten) stattfindet. Um die Gewichtung zwischen Lernen und Verwerten zu ermöglichen, wird eine iterationsabhängige Funktion eingesetzt. Bei der Standardfunktion der Toolbox handelt es sich um eine logarithmische Funktion.

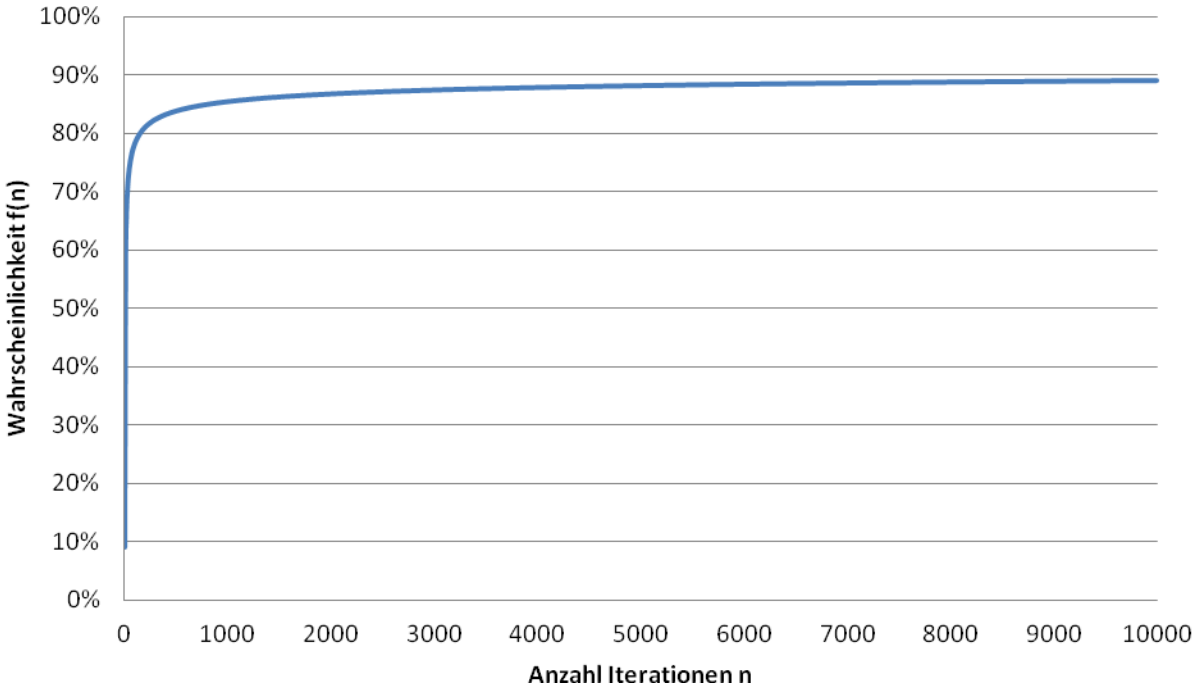



**Diagramm 2: Wahl des Verwertens**

$$f(n) = \frac{1}{\ln(2+n)} \tag{3.2.2}$$

Die Funktion gibt die Wahrscheinlichkeit für die Wahl des Verwertens an und konvergiert gegen 90%. Sollte in einem Zustand noch keine Belohnung enthalten sein, so entfällt die Wahl auf eine rein zufällige Aktion.

Bei der Betrachtung, dass auch andere Funktionen ein ideales Verhältnis zwischen Lernen und Verwerten darstellen könnten, soll zunächst der maximale Kontrast bei-

der Verfahren zur Aktionswahl gegenübergestellt werden, um dessen Auswirkungen kenntlich zu machen.

Im Diagramm 3 wird das Q-Lernen unter der Anwendung der Standardfunktion (blau), dem reinen Lernen (rot) und dem reinen Verwerten (grün) dargestellt. Jeder Kurvenverlauf spiegelt einen Mittelwert von 100 Durchläufen in jedem Iterationsabschnitt wieder.

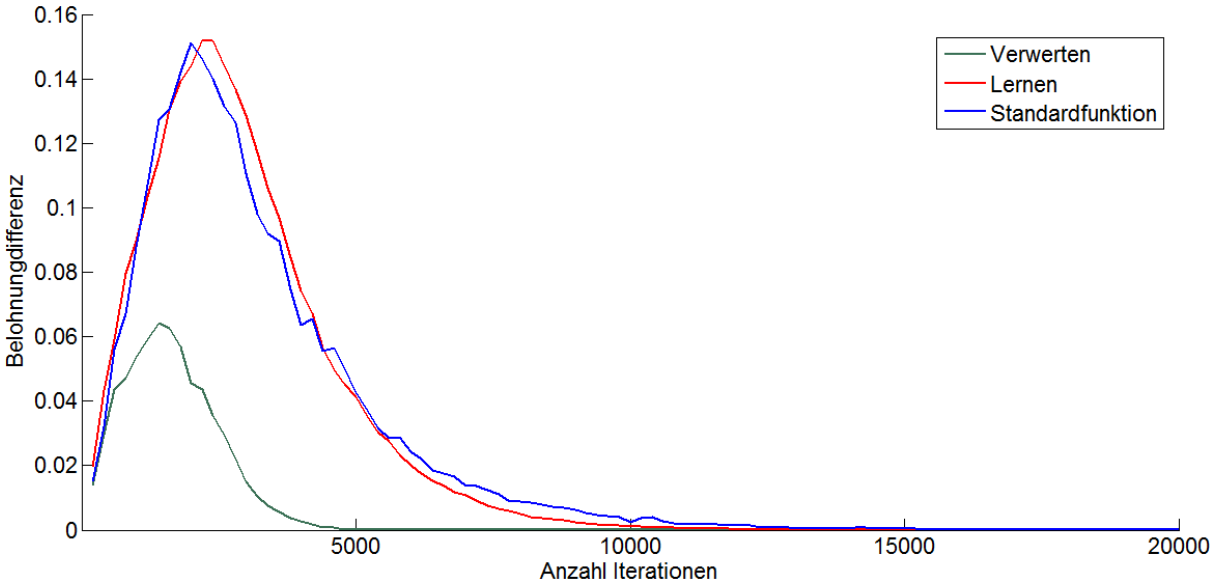



**Diagramm 3: Belohnungsdifferenz der Standardfunktion, dem Verwerten und Lernen**

Unter Anwendung der Standardfunktion ist zu erkennen, dass die Belohnungsverteilung zu Beginn zunimmt und auch bei der Abnahme nicht kontinuierlich fällt. Im Gegenzug zur Wert-Iteration werden hier mehr Iterationsschritte benötigt (N=20000), um eine ausgewogene Verteilung der Belohnung zu erhalten. Das Wertegebirge aus mehreren Durchläufen unterscheidet sich nicht ersichtlich und entspricht dem gleichen Ausgabebild wie in Abbildung 15.

Der grüne Kurvenverlauf stellt die Aktionswahl des reinen Verwertens dar. Es ist zu erkennen, dass sehr frühzeitig eine Konvergenz angestrebt wird. Die geringe Höhe des Kurvenverlaufs lässt auf die schnell festgelegte Strategie, von der nicht mehr abgewichen wird, zurückschließen. Dies liegt daran, dass bei gleicher Ausgangsposition des Zustandes andere Aktionen nie eine Belohnung erhalten können, da diese nie gewählt werden. Die Q-Matrix enthält demnach in jeder Zeile nur eine Wertigkeit. (Siehe Anhang: Tabelle 5)

Aus einem Durchlauf sind in Abbildung 17 der Lösungsweg und das Wertegebirge dargestellt. Ein Unterschied ist zu erkennen. Die Verteilung der Belohnungswerte verlief zunehmend in südlicher Richtung. Demgegenüber bekam der nordöstliche Bereich weniger Belohnungswerte ab.

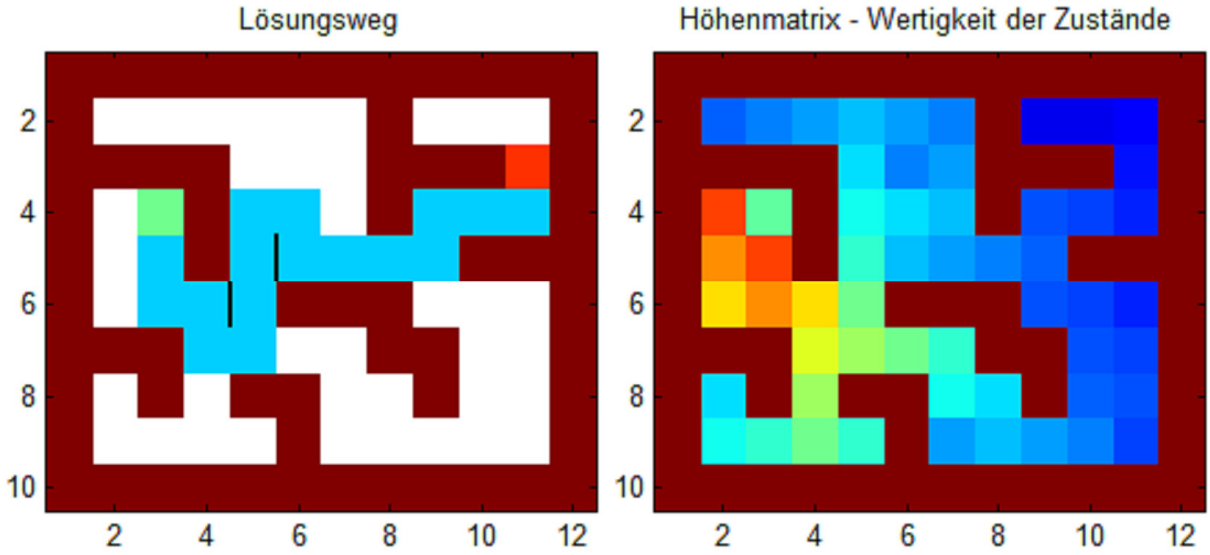



**Abbildung 17: Lösungsweg und Belohnungsverteilung des reinen Verwertens**

Durch das fehlende Einbringen einer zufälligen Aktionswahl, bei bereits verteilter Belohnung, lässt eine einmal entdeckte Strategie keine Abweichung mehr zu. Der damit entstandene Umweg wird als Lösung ausgegeben. Die optimale Strategie wurde nicht aufgedeckt und kann im Laufe weiterer Iterationen nicht mit eingebracht werden.

Das reine Lernen hat annähernd den gleichen Kurvenverlauf wie die Standardfunktion. Nach dem Hochpunkt folgt ein kontinuierlicher Abstieg. Das keine kurzzeitigen Anstiege erfolgen, liegt an der dauerhaften Zufallswahl der Aktionen und dem damit gleichmäßigen Besuch aller Zustände und der Ausübung aller Übergangsfunktionen. Gegenüber der Standardgleichung sinkt der Kurvenverlauf später ab, nähert sich aber dennoch früher der null an. Eine Darstellung der Überlagerung aller vier Richtungsvektoren von jedem Zustand, durch Anwendung der Standardfunktion, dem reinen Verwerten und reinem Lernen befindet sich im Anhang (Abbildung 24 - 26).

Eine probabilistische Aktionswahl nach dem bisher verteilten Belohnungswerten brachte keinen weiteren Erfolg. Es entspricht der gleichen Auswahl, wie dem reinen

Verwerten. Zu Beginn würde eine zufällige Aktion getroffen werden, da alle Belohnungswerte 0 entsprechen und somit die Wahrscheinlichkeit jeder Aktion gleich groß ist. Nach der Ausführung einer Aktion würden im selben Ausgangszustand nur noch drei der vier möglichen Aktionen die Belohnungswertigkeit von 0 besitzen. Die zufällige Wahl nach der Größe der bisherigen Wertverteilung würde demnach immer wieder auf die gleiche Aktion ausfallen, da die übrigen Aktionen keine Belohnungswerte besitzen.

Ein anderer Funktionsverlauf für die Wahrscheinlichkeit des Verwertens soll hier nicht weiter untersucht werden, da Kapitel 3.2.3.4 Q-Initialisierung erheblichen Einfluss auf den Funktionsverlauf zwischen Lernen und Verwerten einnimmt. Auch die probabilistische Aktionswahl in Abhängigkeit der bereits verteilten Belohnungswerte soll dort noch einmal genauer betrachtet werden.

#### *3.2.3.3 Maximale Episodenlänge*

Wie bereits erwähnt, gibt eine Episode eine Zustandsfolge, die der Agent mit der Umwelt einnehmen kann, an. Im Bezug zum Labyrinth (Umgebung) und der Maus (Agent) bedeutet dies eine Strecke, die von der Maus abgelaufen wird. Je größer das Labyrinth ist, desto größer ist der Zustandsraum, den der Agent durchlaufen muss, um eine optimale Strategie zu finden.

Da zu Beginn des Q-Lernens die Zustände noch keine Belohnung besitzen, findet in dem Bereich des Startzustandes eine rein zufällige Wahl der Aktion statt. Unter der Annahme, dass bei unendlich aufeinanderfolgenden Transitionen alle möglichen Aktionen gleich oft gewählt werden, sinkt mit steigender Entfernung der Felder vom Startfeld die Wahrscheinlichkeit, dass diese Zustände eingenommen werden.

Befindet sich zwischen Ziel- und Start-Zustand eine zu große Anzahl an Feldern, so sinkt auch die Wahrscheinlichkeit, dass der Endzustand erreicht wird. Die Verteilung der direkten Belohnung würde demnach wegfallen. Um einen gleichmäßigen Durchlauf aller Zustände zu gewährleisten, wird nach dem Erreichen der maximalen Episodenlänge ein neuer, zufälliger Zustand gewählt. Die maximale Episodenlänge kann

nur erreicht werden, wenn nicht zuvor der Zielzustand erreicht wurde, der zu einem früheren Abbruch führt.

Im Bezug zur Maus im Labyrinth ist dies damit zu interpretieren, dass die Maus eine gewisse Zeit hat, sich in einem Bereich umzusehen. Nach Ablauf eines Zeitraums wird die Maus herausgeholt und an eine neue beliebige Position gesetzt. Nun beginnt der Zeitraum von vorn und die Maus erhält erneut die Möglichkeit sich auf die Suche nach dem Zielzustand zu machen.

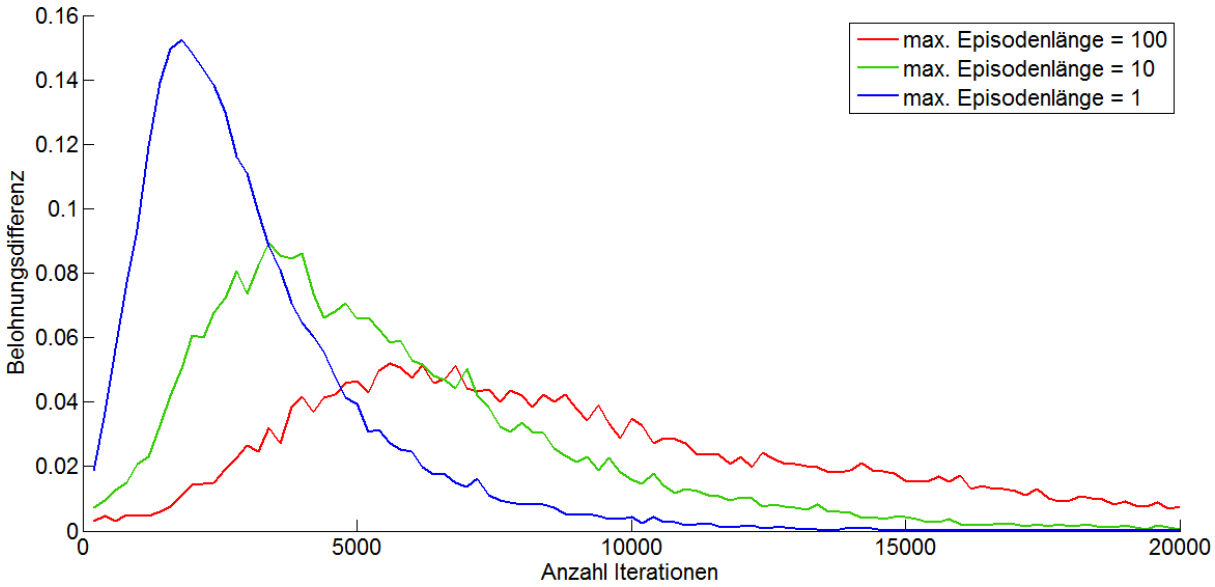



**Diagramm 4: Vergleich verschiedener Episodenlängen**

Diagramm 4 stellt drei Algorithmen mit verschiedenen maximalen Episodenlängen dar. Bei der Anwendung einer kleineren maximalen Episodenlänge findet eine umfangreichere Belohnungsverteilung und schnellere Konvergenz statt. Je kürzer die maximale Episodenlänge ist, desto ausgeglichener ist die Belohnungsverteilung. Eine Darstellung einer einzelnen Iteration aus jeder maximalen Episodenlänge befindet sich im Anhang (siehe Abbildung 27)

#### *3.2.3.4 Q-Initialisierung*

Im bisherigen Algorithmus des Q-Lernens wurde die Q-Matrix mit 0 initialisiert. Zu Beginn des Algorithmus werden ohne direkte Belohnung trotz Transitionen keine Belohnungswerte verteilt und somit auch kein Bezug der Zustände zueinander ermöglicht. Liegen beispielsweise zwischen Ziel- und Startzustand 10 freie Zustandsfelder,

so werden mindestens 10 Transitionen benötigt, damit eine Belohnungswertigkeit am Startzustand angelangt.
Um das Herstellen eines Bezuges der Zustände zueinander frühzeitig zu ermöglichen, wird die Q-Matrix mit 1 initialisiert. Das bedeutet, dass jeder Zustand von Beginn an eine gleichwertige Belohnung besitzt, obwohl keine Transition getätigt wurde.

Durch die vorhandene Belohnungswertigkeit ist zu Beginn jeder Zustand für den Agenten genauso von Interesse wie das eigentliche Ziel. Der Unterschied liegt darin, dass der Zielzustand eine direkte Belohnung verteilt, also dauerhaft die Wertigkeit 1 ausgibt, die zum Ergebnis hinzu addiert wird. Alle anderen Zustände besitzen eine abgeschwächte Belohnung von 1, die im Laufe von Transitionen durch den Diskontierungsfaktor abnimmt.
In der folgenden Tabelle wird das Verhältnis zwischen einer willkürlichen Strategie und der optimalen Strategie bei verschiedenen Einstellungen aus 100 Durchläufen dargestellt.

| | r=1<br>Q=0<br>Standard-funktion | r=1<br>Q=1<br>Standard-funktion | r=1<br>Q=0<br>reines Verwerten | r=1<br>Q=1<br>reines Verwerten | r=10<br>Q=1<br>reines Verwerten | r=1<br>Q=10<br>reines Verwerten | r=1<br>Q=5<br>reines Verwerten |
|---|---|---|---|---|---|---|---|
| Strategie | 0 | 0 | 68 | 38 | 35 | 0 | 0 |
| Optimale Strategie | 100 | 100 | 32 | 62 | 65 | 100 | 100 |

**Tabelle 3: Vergleich von Lösungen unter verschiedenen Parametern**

Als Vergleich befindet sich in der ersten Spalte die bisherige Einstellung der Parameter. Die Standardfunktion bezieht sich auf die Gleichung 3.2.2. Zu erkennen ist, dass bei der Umstellung auf reines Verwerten die Anzahl der optimalen Strategie stark abnimmt.
Bei der Initialisierung der Q-Matrix mit 1 steigt die Anzahl der optimalen Strategien wieder an. Durch weitere Belohnungswerte werden im Laufe des Algorithmus die bereits vorhandenen Belohnungswerte überschrieben (siehe Gleichung 2.2.3). Um dennoch einen Kontrast zum eigentlichen Ziel herzustellen, ist es naheliegend die direkte Belohnung zu erhöhen. Mit einer Erhöhung der direkten Belohnung verlieren jedoch die vorherigen Transitionen mit den initialisierten Q-Werten an Bedeutung. Es würde nahezu dem reinen Verwerten aus Abbildung 17 (bzw. Diagramm 3) entsprechen. Trotz Auffinden einer Lösung bleibt die optimale Strategie oft aus.

Unter der Anwendung des reinen Verwertens bewirkt dieses Verfahren zwei Vorteile. Zum einen wird für den Agenten auch das Interesse in noch nicht besuchte Zustände geweckt. Ein bereits besuchter Zustand, der keine direkte Belohnung ausgibt, verliert durch den Diskontierungsfaktor nach der ausgeführten Übergangsfunktion an Belohnungswertigkeit. Der offene, noch nicht besuchte Zustand, besitzt nach der Transition demgegenüber eine höhere Belohnungswertigkeit. Gelangt der Agent in die gleiche Ausgangsposition, so entscheidet er sich für eine andere Aktion, da die Aktionswahl durch das Verwerten nicht zufällig getroffen wird und die bereits ausgeführte Aktion nicht dem Auswahlkriterium entspricht.

Der zweite Vorteil liegt darin, dass mit Ausnahme vom Anfang des Algorithmus, keine willkürliche Aktionswahl getroffen wird. Jede Aktion wird in jedem Zustand gleich oft getätigt. Demnach ist das Lernen nicht von der Iterationszahl abhängig, wie es in der Gleichung 3.2.1 der Fall war.
Die Auswahl einer zufälligen Aktion erfolgt nur unter den möglichen Nachfolgezuständen, mit gleicher maximaler Belohnungswertigkeit. Zu Beginn des Q-Lernens entspricht dies alle Aktionen. Dies kann auch als eine modifizierte Art des Lernens betrachtet werden, die für jeden Zustand individuell angewendet wird, bis die Auswirkung der vom Ziel ausgehenden Belohnungswertigkeit eintrifft.

In Diagramm 5 werden die Kurvenverläufe mit den 4 verschiedenen Parametereinstellungen, bei denen 100% der optimalen Strategien aufgedeckt wurden, aus der Tabelle 3 dargestellt. Der rote Kurvenverlauf entspricht der bisherigen Einstellung und damit dem blauen Funktionsverlauf aus Diagramm 4.
Durch die alleinige Änderung der Q-Initialisierung ist zu erkennen, dass zum Startzeitpunkt eine höhere Belohnungsverteilung stattfindet (grüner Kurvenverlauf). Der erhöhte Wert lässt aber nicht auf eine bessere Parametereinstellung zurückschließen. Erst durch den Hochpunkt ist zu erkennen, dass sich das Ziel (direkte Belohnung) mehr durchsetzt und eine Festigkeit im System eintritt. Da ein früherer Hochpunkt und auch früherer Abstieg eintritt (gegenüber dem roten Kurvenverlauf), wird eine frühere Konvergenz erreicht, was den Vorteil der Parametereinstellung kenntlich macht.

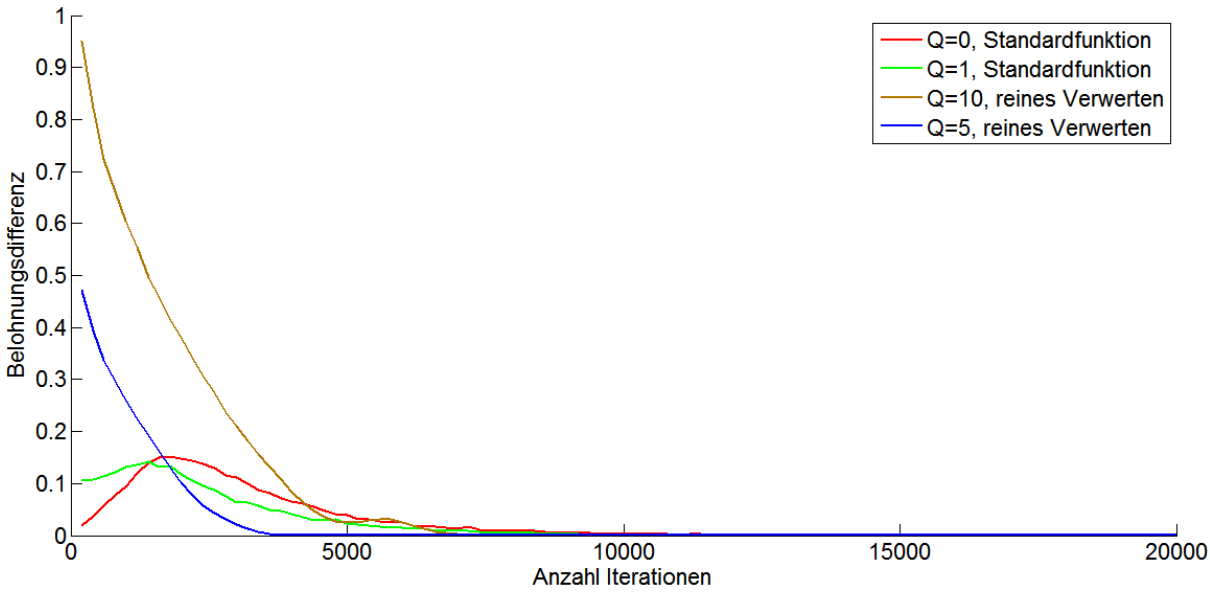



**Diagramm 5: Vergleich des Verlaufs der Belohnungsdifferenz bei verschiedenen Parametern**

Der braune Kurvenverlauf enthält zwei Änderungen. Zum einen die Q-Initialisierung mit der zehnfachen Wertigkeit gegenüber der direkten Belohnung (dem Ziel) und zum anderen wird ausschließlich das Verwerten zur Aktionswahl eingesetzt. Ein Hochpunkt ist durch die große Anzahl an Belohnungswertigkeit nicht zu erkennen. Die Konvergenz benötigt nicht unbedingt länger, denn sie nähert sich zum selben Zeitpunkt den anderen Kurven an und schwankt sogar in diesem Bereich um den roten Kurvenverlauf, wodurch ein Unterschied nicht eindeutig zu erkennen ist.
Bei dem letzten Kurvenverlauf (blau) wird die Q-Matrix mit vierfacher Wertigkeit gegenüber der direkten Belohnung initialisiert. Auch hier ist wie bei dem braunen Kurvenverlauf kein Hochpunkt enthalten. Jedoch konvergiert die blaue Kurve am schnellsten.

Unter allen Parametereinstellungen weist der blaue Kurvenverlauf die optimale Einstellung auf. Es wird eine frühzeitige Konvergenz angestrebt und die optimale Strategie wurde unter allen Durchläufen zu 100% aufgedeckt. Eine geringere Q-Initialisierung als 5 brachte keine 100% optimale Strategie hervor.

Zu der bisherigen optimalen Einstellung der Parameter soll nun noch einmal eine Betrachtung zu der probabilistischen Aktionswahl (am Ende aus Kapitel 3.2.3.2 Entscheidungsfunktion zwischen Lernen und Verwerten) gezogen werden.

Die Q-Matrix ist von Beginn an mit Werten belegt. Auch nach einer ausgeführten Transition und der damit verbundenen neuen Belohnungswertigkeit der gewählten Aktion, enthalten bei erneuter Ausgangsposition auch die übrigen, noch nicht gewählten Aktionen eine Wertigkeit. Die Wahrscheinlichkeit der bereits einmal ausgeübten Aktion betrifft demnach nicht mehr 100% gegenüber einer Initialisierung der Q-Matrix mit 0.

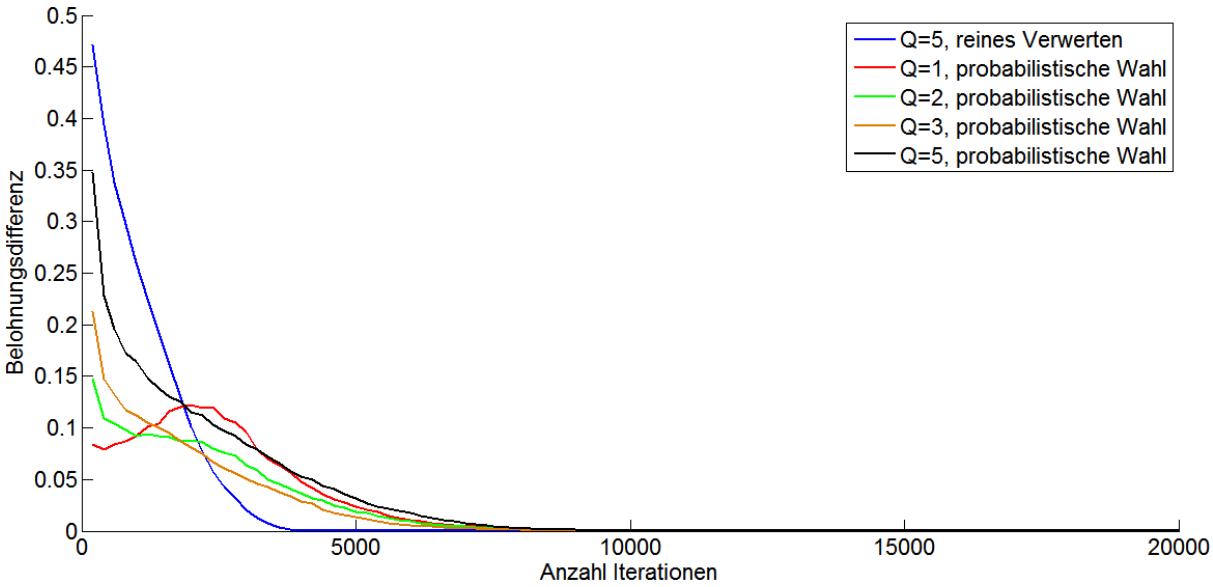



**Diagramm 6: Belohnungsdifferenz bei probabilistischer Wahl der Aktion**

Zum Vergleich dient der vorherige blaue Kurvenverlauf aus Diagramm 5. Zu erkennen ist, dass bei steigender Belohnungswertigkeit für die Q-Matrix, der Kurvenverlauf früher abfällt. Jedoch nur bis zu einer bestimmten Höhe der Belohnungswertigkeit. Ist die Belohnungswertigkeit zu hoch, so benötigt der Algorithmus geringfügig länger für den gesamten Durchlauf, wie bei dem schwarzen Kurvenverlauf zu erkennen ist.

Die bisherige Einstellung des blauen Kurvenverlaufs konnte nicht übertroffen werden. Daher wird eine probabilistische Aktionswahl nicht weiter durchgeführt.

### 3.2.4 Temporal-Difference-Learning

Es wird angenommen, dass durch Einflüsse der Umwelt (wie in Kapitel 2.2 beschrieben) eine Steigerung der Iterationsanzahl benötigt wird, um das gleiche Ziel ohne Umwelteinflüsse zu erreichen.

Eine Abweichung von 20% bedeutet, dass in jeder Transition mit einer Wahrscheinlichkeit von 10% nach links und mit 10% nach rechts von der gewählten Richtung abgewichen wird.

Im nachfolgenden Diagramm wird der Unterschied zwischen einer deterministischen Umgebung (keiner Abweichung) und einer nichtdeterministischen Umgebung (Abweichung von 20%) unter Anwendung der Gleichung 2.1.11 dargestellt.

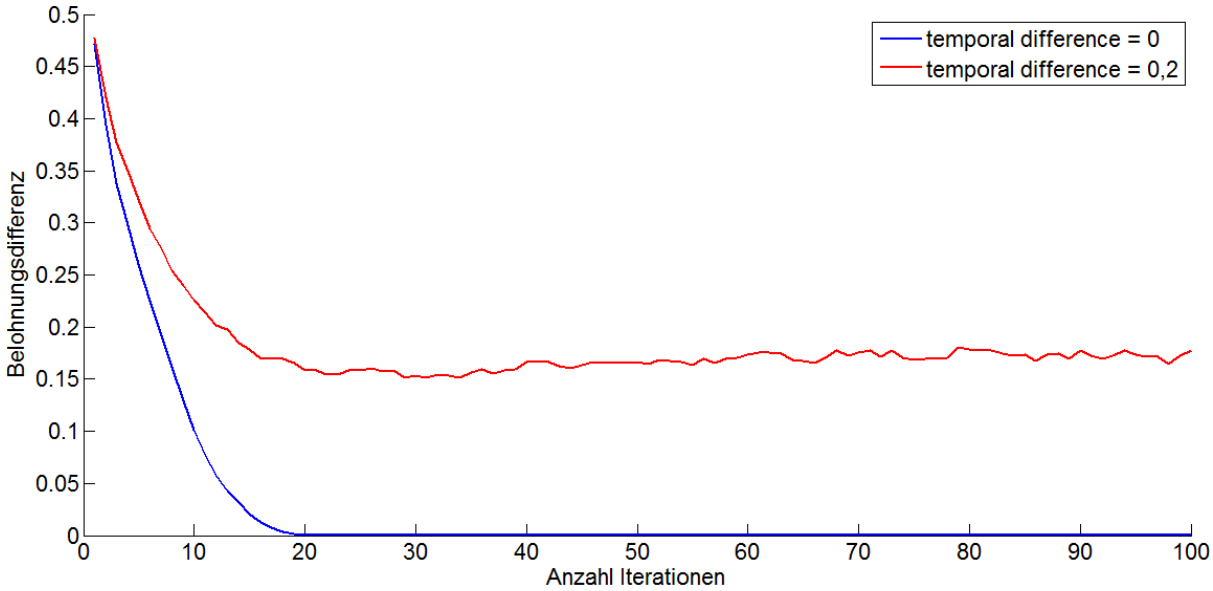



**Diagramm 7: Kurvenverlauf ohne und mit temporal difference**

Bei unveränderter Gleichung (α=1) verhindert eine Abweichung durch schwankende Belohnungswerte die Konvergenzmöglichkeit. Von 100 Durchläufen erhielten 90 eine Strategie, von der lediglich 35 der optimalen Strategie entsprachen.

Eine Erweiterung auf Gleichung 2.2.3 ist daher nötig, um die Abweichungen durch den Gewichtungsfaktor α zu kompensieren. Der Unterschied der Einstellung des Gewichtungsfaktors (Diagramm 8) nach der Gleichung 2.2.2 (blau) und der Standardfunktion aus der Toolbox (rot) soll nun genauer untersucht werden.

Beide Funktionen entsprechen annähernd dem gleichen Verlauf. Zu beachten ist, dass die blaue Funktion sich auf die einzelnen Transitionen bezieht, wodurch der Gewichtungsfaktor auf die Transitionen unabhängig voneinander über die Anzahl der Durchläufe Einfluss nimmt.

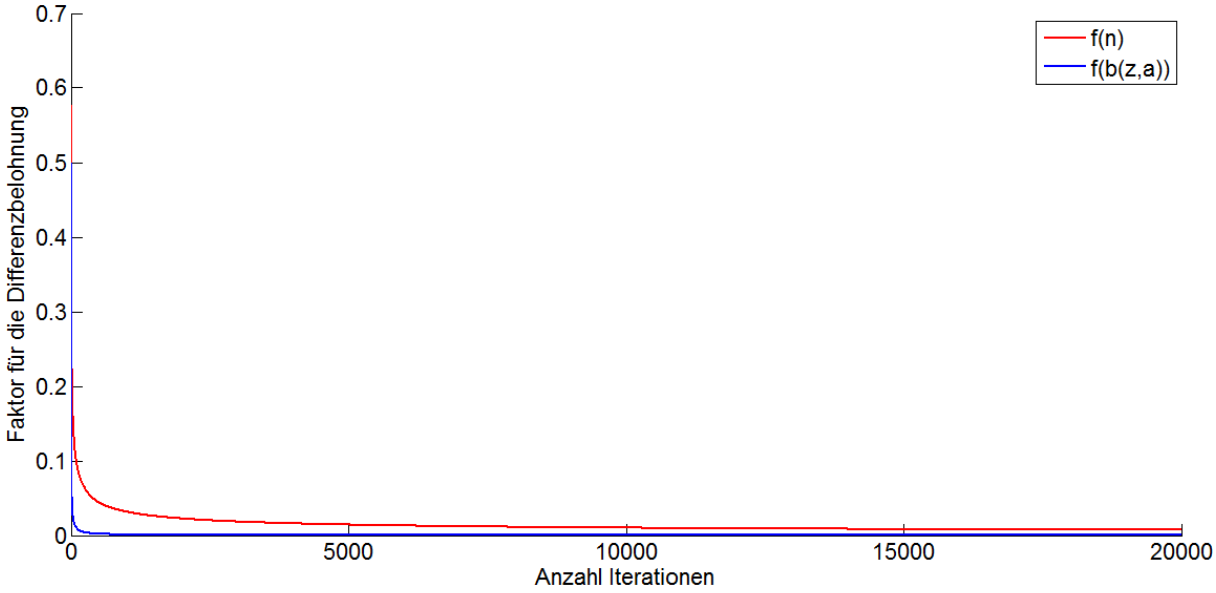



**Diagramm 8: Gewichtungsfaktor α**

$$f(b(z,a)) = \frac{1}{1+b_n(z,a)} \qquad f(n) = \frac{1}{\sqrt{n+2}} \tag{3.2.3}$$

Nach 100 Durchläufen beider Funktionen ergab sich mit den letzten Einstellungen der Parameter (aus Kapitel 3.2.3.4 Q-Initialisierung) keine Lösung. Aus dem Diagramm 9 ist zu erkennen, dass die Konvergenz länger benötigt als bisher. Auch bei erneuter Änderung der Parameter, wie bei der Annahme, dass die Iterationszahl N erhöht werden musste, konnte keine Lösung erzielt werden.

Der Kurvenverlauf fällt durch den Gewichtungsfaktor zu Beginn schneller ab. Es wird angenommen, dass der zu frühe Abfall des Gewichtungsfaktors, der daher zu schnell gegen 0 konvergiert (Diagramm 8), keine Verteilung der Belohnungswerte zulässt.

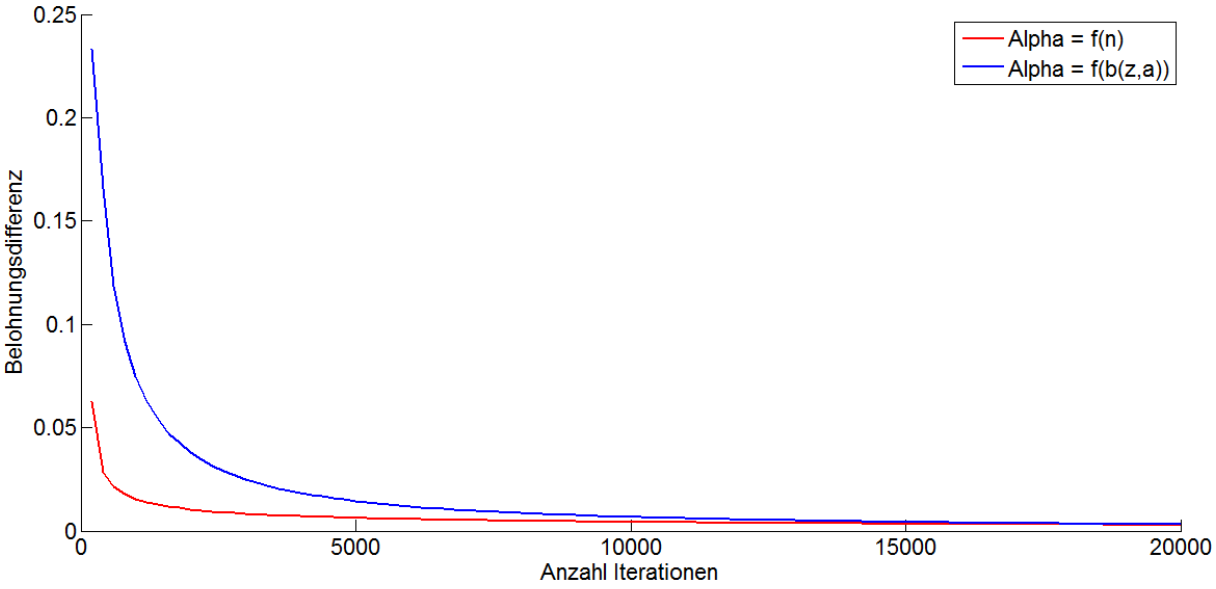



**Diagramm 9: Kurvenverläufe mit Gewichtungsfaktoren**

Eine Funktion des Gewichtungsfaktors mit konstantem negativem Anstieg sollte eine bessere Verteilung der Belohnungswerte ermöglichen (Gleichung 3.2.4 und Diagramm 10). Der Bezug, dass jede Transition einen eigenständigen Gewichtungsfaktor erhält, soll dabei beibehalten werden.

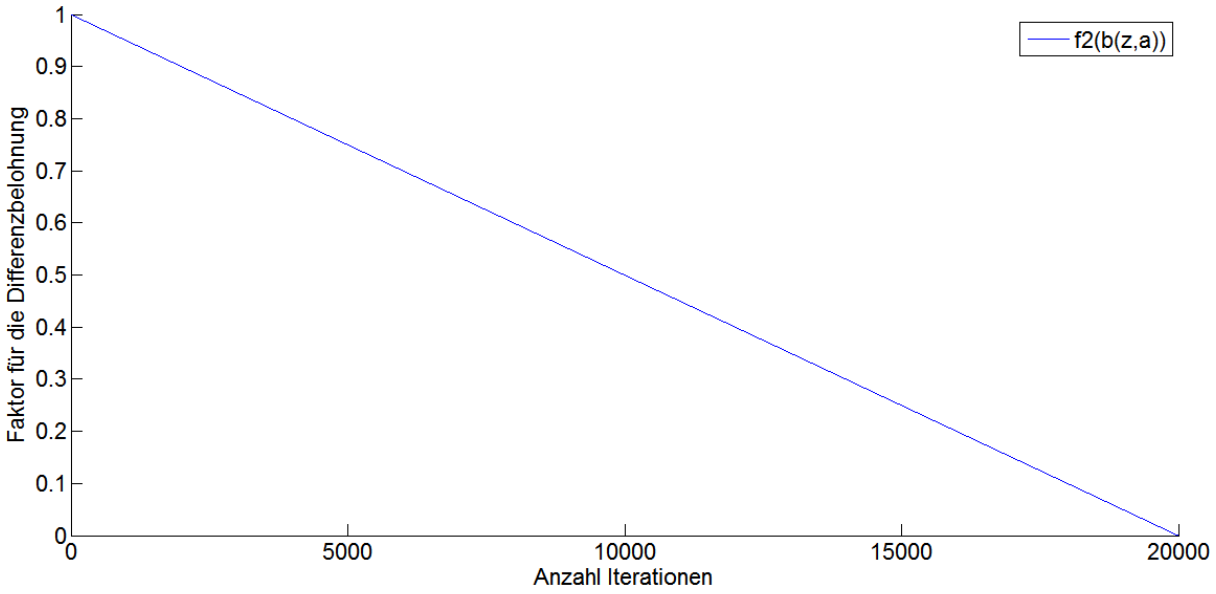



**Diagramm 10: Neuer Gewichtungsfaktor α**

$$f(b(z,a)) = \frac{(1 - b_n(z,a) + N)}{N} \tag{3.2.4}$$

Die rote Funktion aus dem Diagramm 11 stellt mit 73 Strategien und 9 optimalen Strategien aus 100 Durchläufen das Ergebnis dar. Eine Konvergenz kann durch die

geringe Iterationszahl N nicht erreicht werden. Da jede Transition den Gewichtungsfaktor für sich einzeln zählt, muss jede Transition die Anzahl der Gesamtiterationen durchlaufen, um eine Konvergenz zu ermöglichen. In einem Zustandsraum von 56 Zuständen müssten 2,24Mio. Iterationen durchgeführt werden.

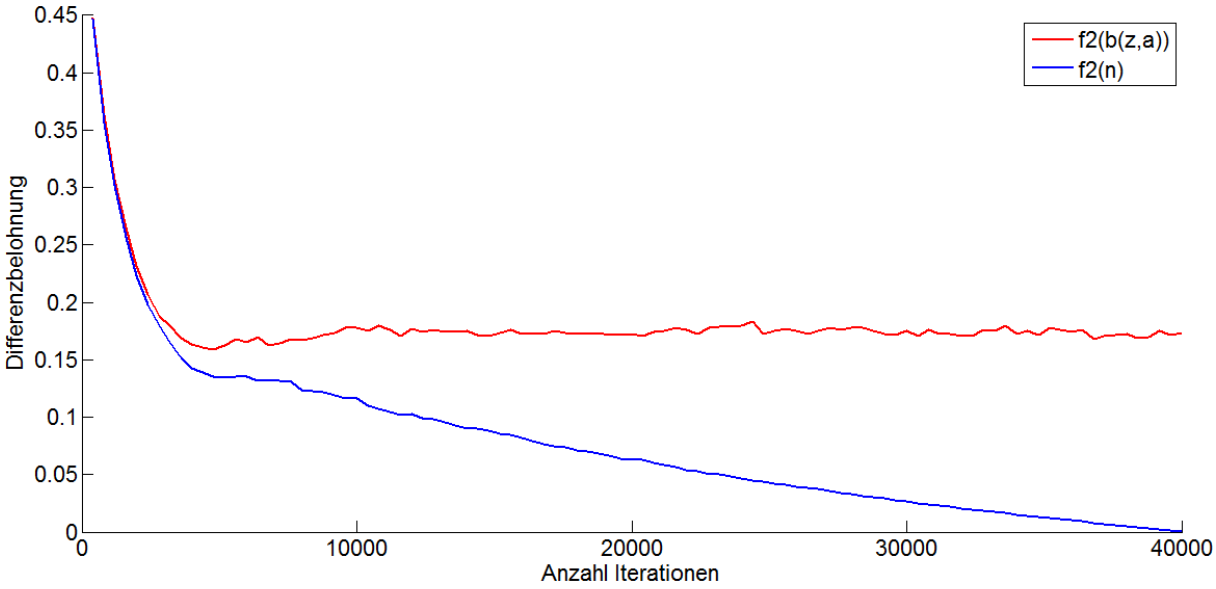



**Diagramm 11: Kurvenverläufe durch neuen Gewichtungsfaktor**

Bezieht sich der Gewichtungsfaktor auf die aktuelle Anzahl der Iterationen (blaue Funktion), so wird frühzeitiger eine Konvergenz angestrebt. Dabei gab es mit 67 Strategien und 33 optimalen Strategien keinen Ausfall.

### 3.2.5 Neumodellierung der Wissensquellen

Bisher fand eine Anpassung der Algorithmen des MDP-Verfahrens mit der Problemstellung der Maus im Labyrinth statt. Bevor der Algorithmus verwendet werden kann, muss zur Vorbereitung eine P-und R-Matrix aus den Labyrinthdaten und den Aktionsmöglichkeiten erstellt werden (Abbildung 18).

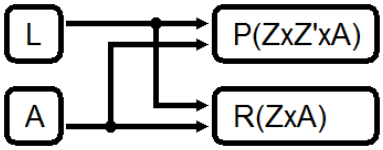



**Abbildung 18: Schema der Vorbereitung zur Erstellung der P-und R-Matrix zur Anwendung des Algorithmus aus der Toolbox**

Die Aktionen sind durch die P-Matrix fest in der Umwelt eingegliedert. Die P-Matrix spiegelt nur die Auswirkung einer gewählten Aktion wieder. Sie dient als eine Tabelle, die vor der Anwendung des Algorithmus aus der Umwelt erstellt wird. Eine veränderte Umwelt muss vollständig neugelernt werden. Eine Anpassung oder Verwendung der vorherigen Q-Werte ist nicht möglich, da bei Änderung der Hindernisse eine Änderung der P-, R-und Q-Matrix zufolge hätte.

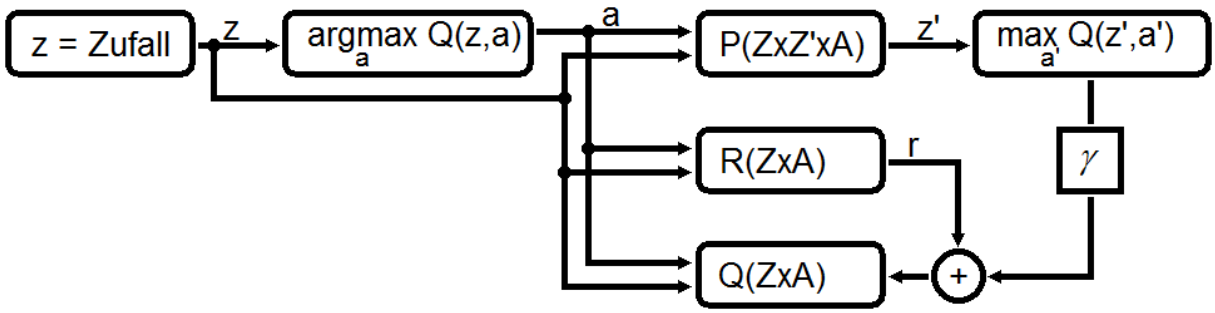



**Abbildung 19: Anwendung der Elemente im Algorithmus aus der Toolbox**

In Abbildung 19 wird der beschriebene Ablauf einer Iteration zum Q-Lernen dargestellt. Die einzelnen Komponenten werden zu der bisherigen Parametereinstellung betrachtet.

Die Bedeutung einer Aktion steht nach bisherigem Ablauf des Programms nicht zur Verfügung. Die Aktionen sind ausschließlich in einer Variablen als Zahl von 1-4 hinterlegt. Eine inhaltliche Bedeutung dieser Zahlen kann nur im Zusammenhang mit den Zuständen und mit Hilfe der P-Matrix wiedergegeben werden.

Die Möglichkeit, das bisher gelernte Wissen bei einer Hindernisänderung im Labyrinth anzuwenden, ist durch die statische P-Matrix nicht gegeben. Eine Umstrukturierung ist daher für das weitere Vorgehen notwendig. Auch bei einer Änderung des Zielfeldes muss eine Anpassung der R-Matrix vorgenommen werden, bevor der Algorithmus fortgesetzt werden kann.

Da die P-Matrix eine statische Abbildung der Welt wieder gibt, die dem Agenten so nicht zur Verfügung steht, wird nun ein Ansatz gewählt, bei dem auf die P-Matrix verzichtet wird.
Die Q-Matrix stellt am Ende des Q-Lernen Algorithmus‘ die Lösung dar. Sie entspricht bisher dem gleichen Aufbau, wie der R-Matrix und somit einer Sortierung aller

möglichen Zuständen (Zeilenvektoren) mit deren Aktionen (Spaltenvektoren). Der Zeilenvektor gibt in allen drei Matrizen (P-, R-und Q-Matrix) immer den aktuellen Zustand an. Daher wird über die eine Variable z (siehe Abbildung 19) auf alle möglichen Elemente zugegriffen, die für den Ablauf des Algorithmus benötigt werden. Um die Sortierung der Zustände aufzulösen, muss eine von einer Variablen z unabhängige Ermittlung des Nachfolgezustandes und der direkten Belohnung erfolgen.
Um auf eine P-Matrix zu verzichten, soll der Nachfolgezustand, unter der Anwendung des aktuellen Zustands und der auszuführenden Aktion, aus den Labyrinthdaten direkt ausgelesen werden. Abbildung 20 gibt eine Gesamtübersicht über die Struktur und deren Bezug zueinander wieder. Zur Vereinfachung wurde die Abfrage nach einem Hindernis aus den Labyrinthdaten nicht dargestellt. Sollte es sich bei dem Nachfolgezustand um ein Hindernis handeln, so wird der Nachfolgezustand dem aktuellen Zustand gleichgesetzt (z‘=z).

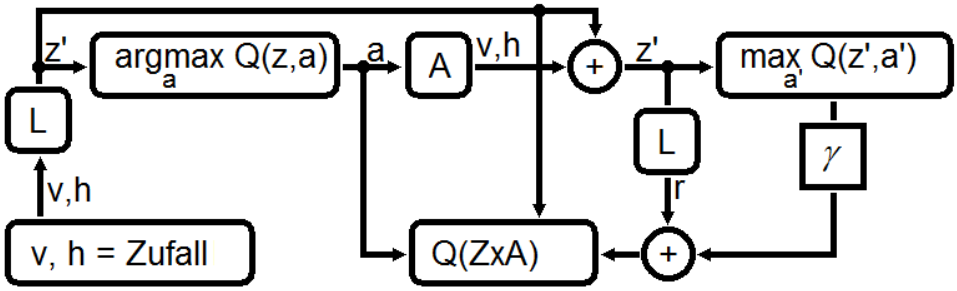



**Abbildung 20: Neue Struktur der Elemente zur Anwendung des Q-Lernens**

Die direkte Belohnung soll durch eine direkte Abfrage, ob es sich bei dem Nachfolgezustand im Labyrinth um das Ziel handelt, erfasst werden. Die bisher vordefinierte R-Matrix wird somit umgangen.

Die Q-Matrix wird mit der Größe und Struktur des Labyrinths erstellt. Auch Felder mit einem Hindernis müssen in der Q-Matrix aufgelistet werden, damit bei einer Änderung der Umwelt (Hindernisse) keine Neuinitialisierung benötigt wird. Die neue Q-Matrix entspricht einer Schablone aus der Struktur des Labyrinths.
Der Vorteil besteht darin, dass mit einer gewählten Aktion der Agent im Labyrinth eine Bewegung ausübt und zugleich diese Aktion genutzt wird, um in der Q-Matrix den entsprechenden Nachfolgezustand zu finden. Da in der bisherigen Q-Matrix der Belohnungswert nach Zustand und Aktion eingeordnet wurde, fällt die Aktion in der neuen Q-Matrix in die dritte Dimension.

Aus der zuvor zweidimensionalen Q-Matrix wird nun eine dreidimensionale Q-Matrix, bei der sich der Zustand nach der vertikalen und horizontalen Ausrichtung orientiert. Die neue Form der Q-Matrix entspricht Q(S(v)xS(h)xA). Der Index nach Zeilen- und Spaltenvektor gibt nun keinen direkten Zustand an. Lediglich der Index für die Tiefe der Matrix orientiert sich nach den Aktionen, die somit eine feste Reihenfolge beibehalten müssen.

Mit der neuen Struktur nach Abbildung 20 kann der Algorithmus des Q-Lernens lediglich in einer deterministischen Umgebung angewendet werden. Eine Wahrscheinlichkeitsbetrachtung über die Abweichung der Transition zum gewünschten Nachfolgezustand ist nicht gegeben.

### 3.2.6 Hindernisänderung in der Umwelt

Eine Konfliktsituation wäre die Anwendung einer Strategie auf eine veränderte Umgebung. In Diagramm 12 sind zwei Lernkurven aus einem Durchlauf dargestellt. Die rote Kurve entspricht dem Lernfortschritt des Agenten zu Beginn des Algorithmus. Nach Ablauf des Algorithmus ist die Q-Matrix mit den Belohnungswerten nach der optimalen Strategie gefüllt.

Der Agent besitzt nun das Wissen über eine feste Umgebung. Wendet er dieses Wissen an und stößt dennoch auf ein Hindernis, so muss zunächst der Ablauf der Strategie gestoppt und eine neue Lernphase eingeleitet werden. Dabei wird die bisher verwendete Q-Matrix weiterhin eingesetzt. An der blauen Kurve (Diagramm 12) ist die zweite Lernphase zu erkennen.

Durch das neu platzierte Hindernis entfällt ein Zustand. Da bereits aus allen anderen Zuständen eine Strategie vorhanden ist, muss nur ein Umweg gelernt werden, bis der Agent an einen Zustand anknüpfen kann, der in einer anderen Strategie enthalten ist, im Gegensatz zu den Strategien, die das neu platzierte Hindernis auf dem Weg zum Ziel einbeziehen. Alle anderen Belohnungswerte bleiben in der Q-Matrix, mit Ausnahme der Zustände, die nun weiter vom Ziel entfernt liegen, erhalten.

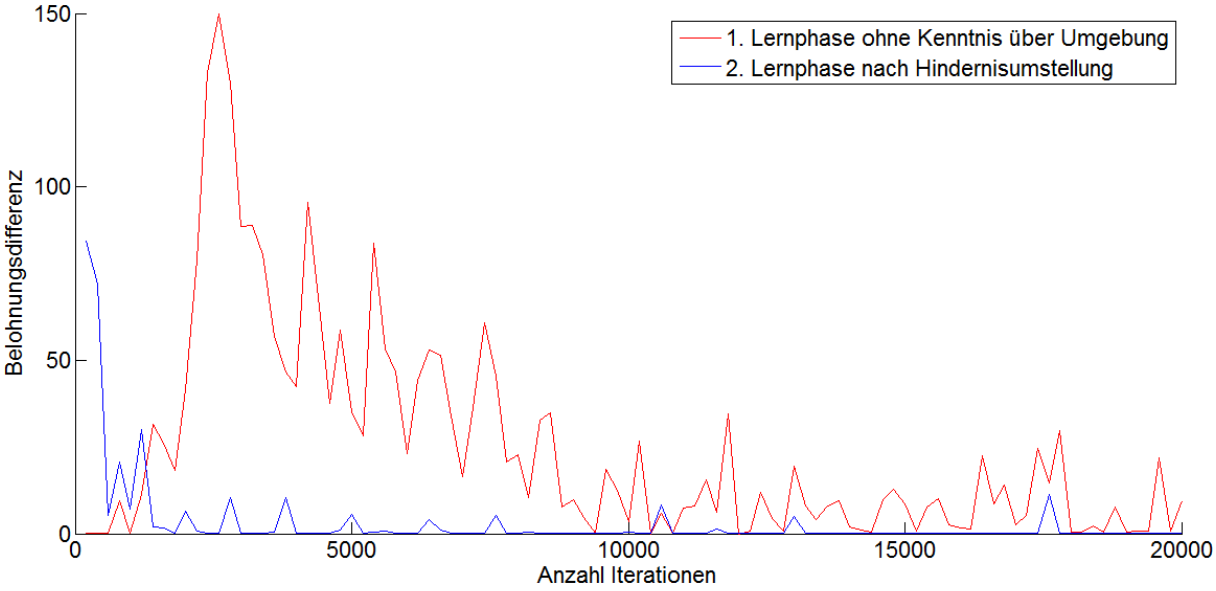



**Diagramm 12:Lernverhalten nach Hindernisumstellung**

In Abbildung 21 sind drei Übersichten über das Labyrinth mit den bisher beschriebenen Geschehnissen dargestellt. In der linken Abbildung befindet sich noch einmal die Ausgangssituation nach der ersten Lernphase. Der Agent kennt die optimale Strategie und kann diese auch bewältigen.

Die Abbildung in der Mitte gibt das Verhalten des Agenten nach der zweiten Lernphase wieder. Der Agent bemerkte ein Hindernis auf seiner Strecke und konnte somit die optimale Strategie nicht mehr anwenden. Nach der erneuten Lernphase konnte die neue optimale Strategie ermittelt werden.

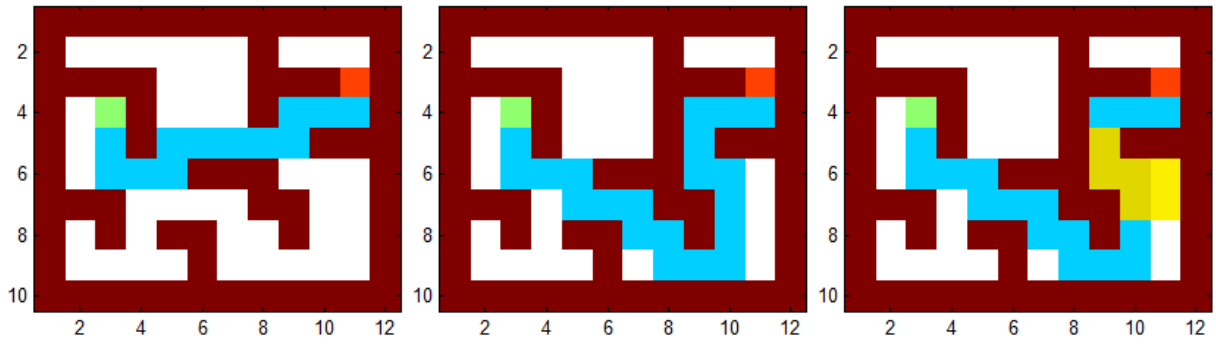



**Abbildung 21: Strategieverlauf bei veränderter Umgebung**

Im letzten Teilbild ist die Änderung, die der Agent mit der zweiten Lernphase durchführen musste, kenntlich gemacht. Die gelb markierten Zustände verwiesen vor der Hindernisplatzierung in dessen Richtung und hätten ohne eine erneute Lernphase

keinen Lösungsweg zum Ziel hervorgebracht. Vier der sechs gelben Zustände befinden sich in der neuen optimalen Strategie.
Die dahinter in Richtung Startzustand befindlichen Zustände haben nun eine höhere Distanz zum Zielzustand und verloren durch den Diskontierungsfaktor an Belohnungswerten. Dies ändert jedoch nichts an ihrer Strategie.

# 4 Bewältigungsverhalten

Das Bewältigungsverhalten (engl. Coping) ist ein psychologisches Verfahren eines Individuums, das zur Bewältigung von Konflikten angewendet wird. Es gibt zwei Arten des Copings, wovon nur das problemorientierte Coping genauer betrachtet werden soll [Sta08].
In dem Bezug zur hier gestellten Aufgabe soll die Erweiterung der Aktionen um NO, SO, SW und NW ermöglicht werden. Diese Aktionen sollen nicht in das Programm implementiert werden. Die Maus soll mit Hilfe einer Bewältigungsstrategie alleine zu diesen Aktionsmöglichkeiten gelangen.

Eine nichtdeterministische Umgebung nach der alten Struktur (Abbildung 19) bewirkte eine Abweichung nach links oder rechts von der gewählten Richtung. In der Realität ist jedoch davon auszugehen, dass eine mögliche Aktion vollzogen wird, die dem Agenten nicht bewusst sein muss. Demnach kann der Agent auch in einer Nebenhimmelsrichtung (z.B. Nordost) landen, obwohl ihm nur die Himmelsrichtungen als Aktionen bekannt sind. Die Nebenhimmelsrichtungen sollen in zukünftigen Aktionen zur Verfügung stehen, wodurch auch eine Betrachtung der Abweichung in diese Richtungen nicht zu vernachlässigen ist.
Da die Nebenhimmelsrichtungen erst erlernt werden sollen, wird eine nichtdeterministische Umgebung außer Betracht gelassen. Es ist anzunehmen, dass die Anfrage nach einer Bewältigungsstrategie nicht eintritt, wenn der Agent durch die Ausführung unbewusster Aktionen sein Ziel erreicht. Der Agent würde nie in einer Konfliktsituation mit der Umgebung und seinem Aktionsrepertoire stehen.

## 4.1 Struktur des Aktionsrepertoires

Das neustrukturierte Modell der Wissensquellen (Abbildung 20) bewirkt den Vorteil, eine neue Struktur des Aktionsrepertoires im Programm zu implementieren. Die Aktionen sind in einer Zellmatrix nach dem Zeilenvektor angeordnet. In der ersten Spalte

befindet sich die Bezeichnung der Aktionen, die dem Tupel aus der zweiten Spalte zugeordnet sind. Im Tupel befinden sich die Werte der Grundaktionen.

Um eine Auseinandersetzung mit einzelnen Aktionen zu ermöglichen, muss der Inhalt jeder Aktion abrufbar sein. Die Bedeutung über die Aktion selbst muss durch den Inhalt erkennbar sein. Der inhaltliche Aufbau der Aktionen besteht aus Grundaktionen, die den einzelnen Aktoren der Maus zugeordnet werden.

Die technische Maus von Theseus besaß eine Aktion zur Drehung und eine weitere Aktion zur Vorwärtsbewegung. Bei der simulierten Maus in Matlab wird keine Drehung, sondern eine direkte Bewegung in entsprechende Richtung ausgeführt.
Es wird angenommen, dass die Maus mit zwei Aktoren ausgestattet ist. Ein Aktor ist für die vertikale Richtung und ein Aktor für die horizontale Richtung ausgelegt. Demnach bewegt sich die Maus ohne eine Drehung zu vollführen durch das Labyrinth, indem sie den entsprechenden Aktor vor oder zurückschaltet.

In der nachfolgenden Abbildung sind auf der linken Seite das Schema der Aktionskombination mit den Beispielen Nord und West sowie auf der rechten Seite der Bezug der gewählten Grundaktionen zur gewählten Wertigkeit dargestellt.

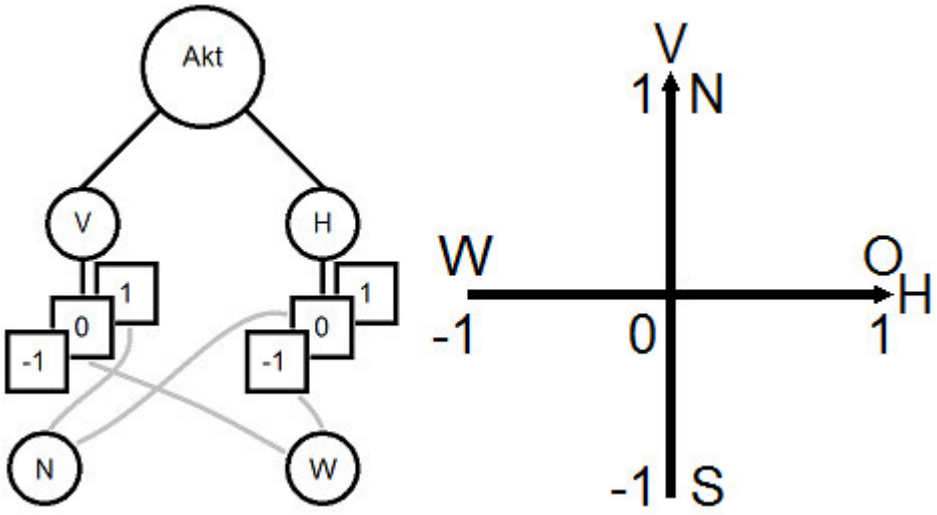



**Abbildung 22: Aktionsaufbau**

Bei der Wahl einer Aktion wird auf die vertikale und horizontale Grundaktion zurückgegriffen. Für den Wert 1 wird dem entsprechenden Aktor die Vorwärtsbewegung

zugewiesen und für den Wert -1 die Rückwärtsbewegung. Bei dem Wert 0 ist der Aktor ausgeschaltet.
Für die bisherigen Aktionen von Nord, Ost, Süd und West ist immer ein Aktor aktiv. Dennoch werden die Werte beider Aktoren übermittelt. Eine Aktion besteht aus zwei Werten, nach dem Schema Aktion = {V, H}, z.B. für die Aktion Nord = {1, 0}.

Nach dem Schema aus Abbildung 22 besteht jede Aktion aus einem Tupel. Jeder Tupel besitzt eine andere Wertigkeit, wodurch der Inhalt über die entsprechend gewählte Aktion Aufschluss gibt. Die vorhandenen Aktionen können nun miteinander verglichen werden, da ihre Bedeutung nicht allein durch eine Ziffer gekennzeichnet ist.

## 4.2 Erweiterung des Aktionsrepertoires

Durch eine umstrukturierte Umwelt kann es vorkommen, dass der Agent den Zielzustand mit den vorhandenen Aktionen nicht erreicht. Bisher wurden Aktionen, die sich nach den Himmelsrichtungen orientierten, eingesetzt. Der Agent konnte keine Transition zu einem Zustand, der sich diagonal zu ihm befand, ausüben, da die entsprechenden Aktionen der Nebenhimmelsrichtungen fehlten.

Zur Ermittlung der Nebenhimmelsrichtungen werden Überlagerungen der aktuellen Himmelsrichtungen durchgeführt. Jeder Tupel einer Aktion wird mit den übrigen Tupeln kombiniert. Das Ergebnis wird mit der Zellmatrix verglichen. Befindet sich der Tupel noch nicht in der Zellmatrix, so wird dieser als eine neue Aktion hinzugefügt. Um die neue Aktion zu vervollständigen fehlt noch eine Bezeichnung. Diese wird ebenfalls aus der Kombination der bisherigen Bezeichnungen ermittelt.

Zu Beginn besitzt der Agent die Aktionen mit den dazugehörigen Tupeln in der Reihenfolge N(1; 0), O(0; 1), S(-1; 0) und W(0; -1) (Siehe Abbildung 22). Es wird der Reihenfolge nach jede Aktion mit den nachfolgenden Aktionen verglichen. Die erste Kombination erfolgt von Nord und Ost. Daraus resultiert die Aktion NO(1; 1) mit einer korrekten Bezeichnung.

Bei der zweiten Kombination, die aus Nord und Süd erfolgt, resultiert die Aktion NS(0; 0). Diese Aktion besitzt eine fälschliche Bezeichnung. „Stillstand“ würde eine verständlichere Bezeichnung wiedergeben. Zudem bringt diese Aktion dem Agenten für die hier betrachtete Welt nicht weiter. Es existiert keine Situation, die dem Agenten einen Vorteil für die Aktion „Stillstand“ bringen würde.

Bei der Kombination der Aktionen Ost und Süd entsteht mit OS(-1; 1) ebenfalls eine falsche Bezeichnung. Da in der Semantik die vertikalen Aktionen eine höhere Bedeutung tragen, kann durch eine Änderung der Reihenfolge der Aktionen eine korrekte Bezeichnung ermittelt werden.
Die Zellmatrix für die Ausgangsaktionen wird nun in der Reihenfolge der vertikalen und anschließend der horizontalen Aktionen angeordnet (N, S, O, W).

Wird zu dem bereits platzierten Hindernis aus Abbildung 21 ein weiteres Hindernis so hinzugefügt, dass der Agent auf die neuen Aktionsmöglichkeiten zurückgreifen muss, entsteht folgende Abbildung:

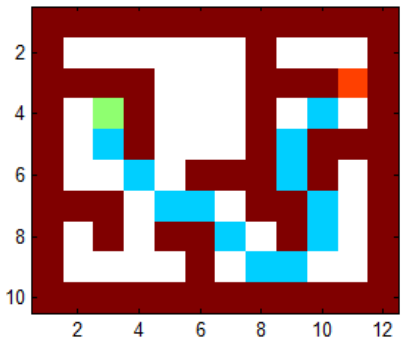



**Abbildung 23: Strategieverlauf nach dem Coping**

Im nachfolgenden Diagramm stellt die rote Funktion die erste Lernphase dar. Zu Beginn ist ein Anstieg vorhanden, da die Belohnungswerte vom Ziel aus erst über mehrere Zustände verteilt wurden. Anschließend festigen sich die Belohnungswerte wodurch die Kurve gegen null konvergiert.

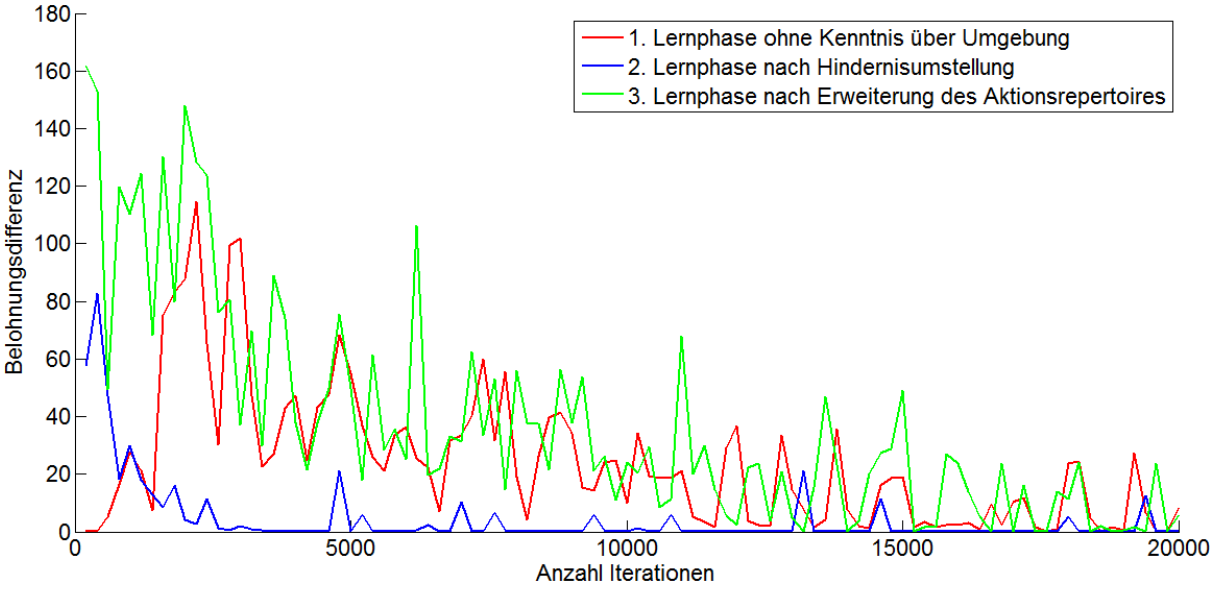



**Diagramm 13: Lernverhalten nach Hindernisumstellung und Bewältigungsverhalten**

Die blaue Funktion stellt eine erneute Lernphase nach der Platzierung eines Hindernisses dar. Sie gibt die Änderung der Belohnungsverteilung einiger Zustände (siehe Abbildung 21, gelbe Zustände) an. Die Q-Matrix wurde bereits vorher mit Werten gefüllt, daher ist lediglich eine Anpassung einiger Werte nötig. Ein Großteil der bereits vorhandenen Werte bleibt wie zuvor erhalten. Somit ergibt sich ein kleinerer Kurvenverlauf gegenüber der roten Funktion.
Die dritte Lernphase ist grün dargestellt und wurde nach der Erweiterung des Aktionsrepertoires durchgeführt. Die erneut hohe Verteilung der Belohnungswerte ist auf die Erweiterung der Q-Matrix zurückzuführen, da diese in der dritten Dimension durch die hinzugefügten Aktionen erweitert wurde (siehe Abbildung 20).

Aus dem Diagramm ist zu erkennen, dass eine erneute Lernphase mit gleichem Aktionsrepertoire eine geringere Änderung der Belohnungsverteilung bewirkt (blaue Funktion), als eine erneute Lernphase mit neuem Aktionsrepertoire (grüne Funktion).

Ein idealer Coping-Algorithmus würde das Aktionsrepertoire lediglich um die Aktionen erweitern, die zur Lösung des Problems benötigt werden. Somit muss ein Zusammenspiel zwischen den Aktionen und der Umwelt stattfinden.

# 5 Zusammenfassung und Fazit

Bei dem Markov-Entscheidungsprozess (MDP) handelt es sich um ein bewertendes Lernverfahren, das unter anderem Zustände, Aktionen und Transitionen einsetzt, um für den Agenten eine optimale Strategie für das Verhalten in seiner Umwelt zu ermitteln. Die optimale Strategie stellt dabei eine Handlungsfolge der Aktionen des Agenten in Abhängigkeit vom Zustand dar.
Das Q-Lernen ist eine Erweiterung der Wert-Iteration. Mit dem Einbringen von Aktionen in die Lösungsmatrix bringt es den Vorteil eine Strategie zu entwickeln, die unabhängig von Weltformeln ist, auf die der Agent zur Anwendung der Strategie nun nicht mehr zurückgreifen braucht. Auf die Entscheidung, welche Methode zur Aktionswahl (Lernen oder Verwerten) bei der Lernphase eingesetzt wird, konnte auch ohne Anwendung einer iterationsabhängigen Funktion Einfluss genommen werden. Dies wurde über vorgetäuschtem Scheinwissen in der Lösungsmatrix ermöglicht, bei dem eine Kombination aus Lernen und Verwerten auf alle Zustände individuell angewendet wurde.
In einer nichtdeterministischen Umgebung erfolgt eine Erweiterung der Gleichung, um die Stabilität des Algorithmus zu gewährleisten. Der Agent benötigt durch unregelmäßige Abweichungen bei Transitionen länger, um sich mit der Umwelt vertraut zu machen. Unter anderem liegt dies auch an der Vielzahl an Parametern, die eine ausführliche Einstellung benötigen, um für die gegebene Problemstellung (Eine Maus sucht im Labyrinth den Käse) den Algorithmus effektiv auszunutzen. Die Suche nach einer geeigneten Konvergenzabfrage als Abbruchkriterium und einem Gewichtungsfaktor zur Verbesserung des Algorithmus bietet hier einen Ansatz zur weiteren Untersuchung, da zuletzt nur 33 optimale Strategien von 100 Durchläufen aufgedeckt wurden.

Zur Anwendung des Algorithmus wurde das Rechenprogramm Matlab mit einer entsprechend vorgegebenen MDP-Toolbox verwendet. Aufgrund statischer P-und R-Matrizen, bei denen die Aktionen fester Bestandteil der Welt und nicht des Agenten waren, wurde ein neustrukturierter Ansatz benötigt. Mit dieser Programmstruktur war

es nun möglich Hindernisänderungen in der Umwelt vorzunehmen und das Aktionsrepertoire mit den Grundaktionen der Aktoren wiederzugeben.
Eine erneute Lernphase nach einer veränderten Umwelt fiel deutlich geringer aus, da bereits Vorwissen vorhanden war und nicht erneut erlernt werden musste. Lediglich einzelne Zustände mussten angepasst werden, da ihre vorherige Strategie nicht zu verwenden war. Die Entscheidung der zu wählenden Aktion fiel immer nach dem Argument des Maximums aus. Eine mögliche Betrachtung über das Argument des zweiten Maximums könnte bei einer Konfliktsituation (mit einem Hindernis) einen Ansatz zur Aufdeckung einer alternativen Strategie geben, ohne eine zweite Lernphase durchzuführen.

Erst die Erstellung einer Zellmatrix zur Strukturierung des Aktionsrepertoires ermöglichte einen Ansatz zum Coping. In der Zellmatrix war jede Aktion nach den Grundaktionen der Aktoren aufgebaut. Eine einfache Kombination der Aktionen ermöglichte die Erweiterung des Aktionsrepertoires. Die Bezeichnung neuer Aktionen resultierte ebenfalls aus der Kombination vorheriger Aktionen. Dabei fiel auf, dass eine nichtgeläufige Bezeichnung wie „Ostsüd“ zustande kam. Erst durch die Festlegung einer höheren Priorität, der Bezeichnungen von vertikalen Aktionen gegenüber den horizontalen Aktionen, brachte die korrekte Bezeichnung von „Südost“. Demnach erfolgte eine neue Anordnung der Aktionen in der Zellmatrix nach Nord, Süd, Ost, West. Einfluss auf den Algorithmus hat diese Änderung nicht. Als eine spezielle Ausnahme könnte die Aktion des Stillstandes betrachtet werden, die durch die Kombination von Nord und Süd zustande kam.
Auch wenn die Erweiterung des Aktionsrepertoires ermöglicht wurde, entspricht dies nicht dem Idealfall einer Copingstrategie. Bei einsetzendem Coping, wurden alle möglichen Aktionen ermittelt. Realistischer wäre die Betrachtung, dass nur die Aktionen ermittelt werden, die zur Lösung des Problems beitragen.
Die Untersuchung einer semantischen Schlussfolgerung, durch vorgegebene Nebenhimmelsrichtungen auf die Bezeichnung von Himmelrichtungen zu gelangen, bietet hier einen weiteren Ansatz, der genauer betrachtet werden kann.

## Literaturverzeichnis


[Beu14] Beurich, Johann (DorFuchs): Was ist $0^0$
Abgerufen am 15.08.2014
URL: https://www.youtube.com/watch?v=mjLF3xlQhYY

[Cha14] Ladine Chadès et al: MDP – Toolbox PDF, Quick Start: Resolving a Markov decision process problem using the MDPtoolbox in Matlab, Januar 2014
Abgerufen am 04.08.2014
URL: http://www.mathworks.com/matlabcentral/fileexchange/25786-markov-decision-processes--mdp--toolbox

[Ert09] Ertel, Wolfgang: Grundkurs Künstliche Intelligenz, 2. Überarbeitete Auflage 2009, Vieweg+Teubner Verlag
ISBN: 978-3-8348-0783-0

[Hog] Hoggett, Reuben: 1952 – „Theseus“ Maze-Solving Mouse – Claude Shannon (American)
Abgerufen am: 19.08.2014
URL: http://cyberneticzoo.com/mazesolvers/1952-%E2%80%93-theseus-maze-solving-mouse-%E2%80%93-claude-shannon-american/

[RN12] Russell, Stuart; Norvig, Peter: Künstliche Intelligenz – Ein moderner Ansatz, 3. Überarbeitete Auflage 2012, Pearson Verlag
ISBN: 978-3-86894-098-5

[Röm14] Römer, Ronald: Übungsskript Kognitive Systeme; BTU-Cottbus-Senftenberg; Wintersemester 2013/14

[SF] Schmidt, Thorsten; Fuchs, David: Einführung in die Künstliche Intelligenz, Universität Tübingen – Geographisches Institut
Abgerufen am: 15.08.2014
URL: http://www.geosimulation.de/methoden/einfuehrung_k

[Sta08] Stangl, Werner: Psychologie News – Coping
Erstellt am: 20.01.2008
Abgerufen am: 19.11.2014
URL: http://psychologie-news.stangl.eu/85/coping
Erste Quelle: Häcker, H. & Stapf, K. (1994). Dorsch Psychologisches Wörterbuch. Bern: Hans Huber
Heckhausen, H. (1989). Motivation und Handeln. München: Springer.
Peters, Uwe Henrik (2000). Psychiatrie, Psychotherapie, Medizinische Psychologie. Urban und Fischer Verlag.

[The] Thewalt, Oliver: EDV Lexikon – Künstliche Intelligenz, Neogrid – Gemeinnütziger Verein zur Förderung und Verbreitung von Kunst und Kultur sowie Volks- und Berufsbildung im öffentlichen Leben e.V. Abgerufen am: 15.08.2014
URL: http://www.neogrid.de/was-ist/K%C3%BCnstliche-Intelligenz

[Wol14] Wolff, Matthias: Vorlesungsskript Kognitive Systeme; BTU-Cottbus-Senftenberg; Wintersemester 2013/14

# Anhang

| Aktion a=Nord | | Nachfolgezustand z' | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Zustand z | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | 5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 6 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 7 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

| Aktion a=Ost | | Nachfolgezustand z' | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Zustand z | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| | 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | 6 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Aktion a=Süd | | Nachfolgezustand z' | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Zustand z | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | 5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 6 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Aktion a=West | | Nachfolgezustand z' | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Zustand z | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 3 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| | 4 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 5 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 6 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Tabelle 4: Komplette P-Matrix nach Labyrinth aus Abbildung 13**

| Q-Matrix | | Aktion a | | | |
|---|---|---|---|---|---|
| | | Nord | Ost | Süd | West |
| Zustand z | 1 | 0 | 0,97527377 | 0 | 0 |
| | 2 | 0 | 1,08363753 | 0 | 0 |
| | 3 | 0 | 1,2040417 | 0 | 0 |
| | 4 | 0 | 1,33782411 | 0 | 0 |
| | 5 | 0 | 1,48647123 | 0 | 0 |
| | 6 | 0 | 0 | 1,6516347 | 0 |
| | 7 | 0 | 0,71097457 | 0 | 0 |
| | 8 | 0 | 0,78997174 | 0 | 0 |
| | 9 | 0 | 0 | 0,8777464 | 0 |
| | 10 | 0 | 1,48647123 | 0 | 0 |
| | 11 | 0 | 1,6516347 | 0 | 0 |
| | 12 | 0 | 0 | 1,83514966 | 0 |
| | 13 | 0 | 0 | 0,97527377 | 0 |
| | 14 | 0 | 5,26315787 | 0 | 0 |
| | 15 | 0 | 0 | 4,73684208 | 0 |
| | 16 | 0 | 0 | 2,51735208 | 0 |
| | 17 | 0 | 0 | 0 | 2,26561687 |
| | 18 | 0 | 0 | 0 | 2,03905518 |
| | 19 | 0 | 0 | 1,33782411 | 0 |
| | 20 | 0 | 0 | 0 | 1,2040417 |
| | 21 | 0 | 0 | 0 | 1,08363753 |
| | 22 | 0 | 4,73684208 | 0 | 0 |
| | 23 | 5,26315787 | 0 | 0 | 0 |
| | 24 | 0 | 0 | 2,79705787 | 0 |
| | 25 | 2,03905519 | 0 | 0 | 0 |
| | 26 | 0 | 0 | 0 | 1,83514967 |
| | 27 | 0 | 0 | 0 | 1,6516347 |
| | 28 | 0 | 0 | 0 | 1,48647123 |
| | 29 | 4,26315787 | 0 | 0 | 0 |
| | 30 | 4,73684208 | 0 | 0 | 0 |
| | 31 | 0 | 0 | 0 | 4,26315787 |
| | 32 | 0 | 0 | 3,10784208 | 0 |
| | 33 | 1,33782411 | 0 | 0 | 0 |
| | 34 | 0 | 0 | 0 | 1,2040417 |
| | 35 | 0 | 0 | 0 | 1,08363753 |
| | 36 | 3,83684208 | 0 | 0 | 0 |
| | 37 | 0 | 0 | 0 | 3,45315787 |
| | 38 | 0 | 0 | 0 | 3,10784209 |
| | 39 | 0 | 0 | 0 | 2,79705787 |
| | 40 | 1,08363753 | 0 | 0 | 0 |
| | 41 | 0 | 0 | 0 | 0,97527377 |
| | 42 | 0 | 0 | 2,26561687 | 0 |
| | 43 | 3,45315787 | 0 | 0 | 0 |
| | 44 | 2,51735208 | 0 | 0 | 0 |
| | 45 | 0 | 0 | 0 | 2,26561687 |
| | 46 | 0,97527377 | 0 | 0 | 0 |
| | 47 | 0 | 0 | 0 | 0,8777464 |

| | | | | | |
|---|---|---|---|---|---|
| | 48 | 0 | 2,51735209 | 0 | 0 |
| | 49 | 0 | 2,79705788 | 0 | 0 |
| | 50 | 3,10784209 | 0 | 0 | 0 |
| | 51 | 0 | 0 | 0 | 2,79705787 |
| | 52 | 0 | 1,83514967 | 0 | 0 |
| | 53 | 2,03905519 | 0 | 0 | 0 |
| | 54 | 0 | 0 | 0 | 1,83514967 |
| | 55 | 0 | 0,71097458 | 0 | 0 |
| | 56 | 0,78997176 | 0 | 0 | 0 |

**Tabelle 5: Q-Matrix unter reinem Verwerten**

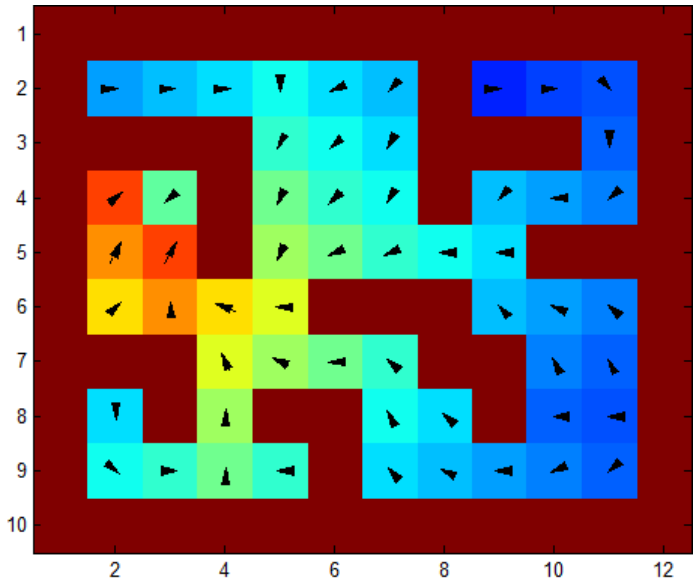



**Abbildung 24: Wertegebirge mit Vektorüberlagerung bei Anwendung der Standardfunktion**

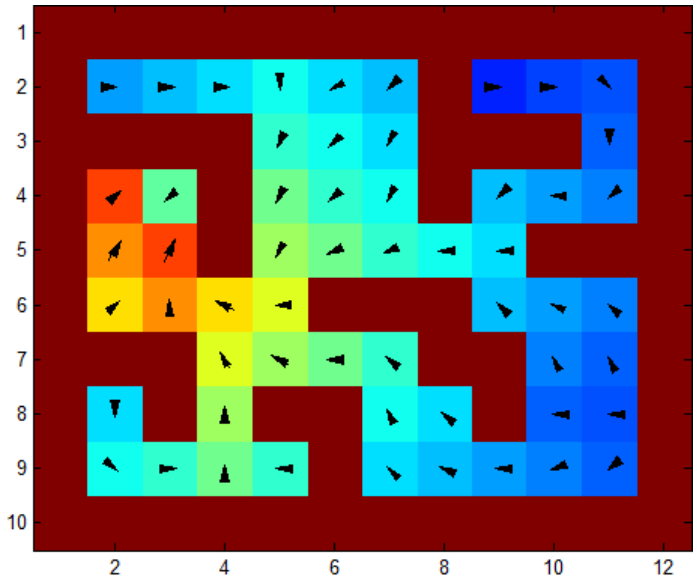



**Abbildung 25:Wertegebirge mit Vektorüberlagerung bei reinem Lernen**

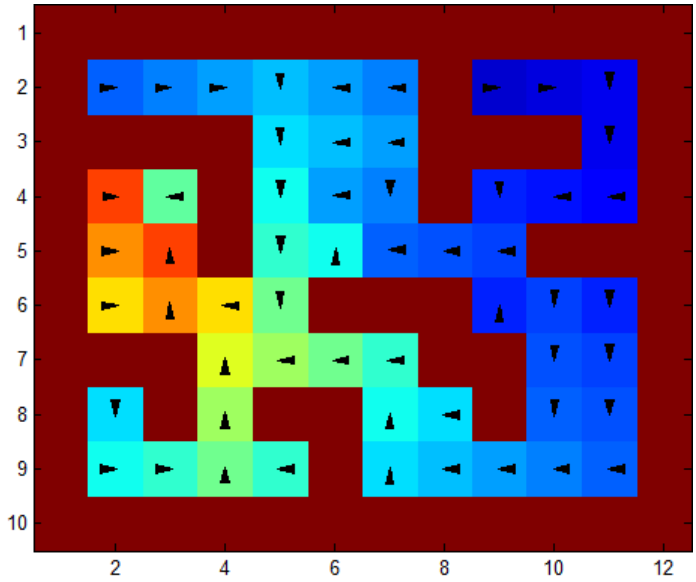



**Abbildung 26: Wertegebirge mit Vektorüberlagerung bei reinem verwerten**

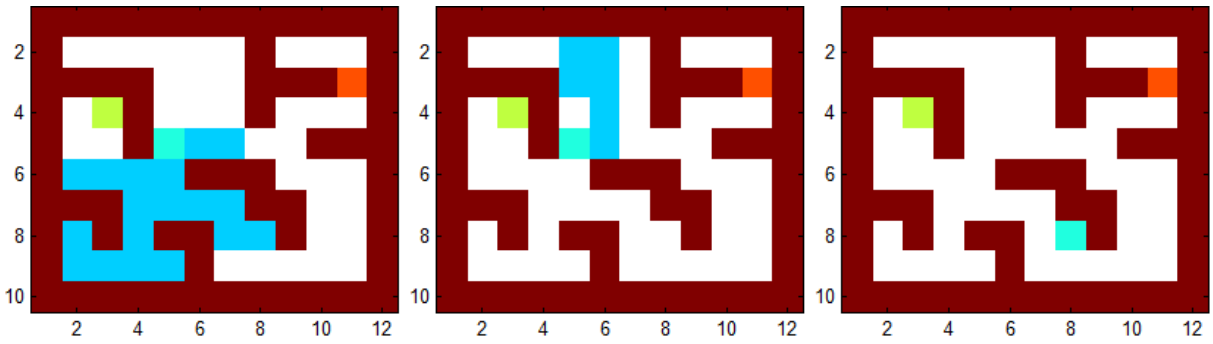



**Abbildung 27: Einzelne Iteration mit unterschiedlicher max. Episodenlänge (v.l.n.r. 100, 10, 1)**