Technology
Arts Sciences
TH Köln

# Untersuchung des Spiel- und Lernerfolgs künstlicher Intelligenzen für ein nichtdeterministisches Spiel mit imperfekten Informationen

Blackjack in der Game-Learning-Umgebung
'General Board Game' (GBG)

Bachelorarbeit

ausgearbeitet von

Simon Meißner

zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

vorgelegt an der

Technischen Hochschule Köln
Campus Gummersbach
Fakultät für Informatik und
Ingenieurwissenschaften

im Studiengang
Medieninformatik

| | |
|---|---|
| Erster Prüfer/in: | Prof. Dr. rer. nat. Wolfgang Konen<br>Technische Hochschule Köln |
| Zweiter Prüfer/in: | Prof. Dr. Matthias Böhmer<br>Technische Hochschule Köln |

Gummersbach, im April 2021

**Adressen:** Simon Meißner
Olgastraße 14
34119 Kassel
smnmnr@gmail.com

Prof. Dr. rer. nat. Wolfgang Konen
Technische Hochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
wolfgang.konen@th-koeln.de

Prof. Dr. Matthias Böhmer
Technische Hochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
matthias.boehmer@th-koeln.de

# Kurzfassung

Die vorliegende Bachelorarbeit untersucht den Spiel- und Lernerfolg verschiedener Reinforcement Agenten für ein rundenbasiertes, nichtdeterministisches Spiel mit imperfekten Informationen, anhand von Blackjack. Dafür wurde im Vorfeld dieser Arbeit das Game-Learning-Framework General Board Game(GBG), welches auf das Lernen beliebiger zugbasierter Spiele durch künstliche Intelligenzen spezialisiert ist, um eine Blackjack Umgebung erweitert. Die vom GBG bereitgestellten Agenten Monte Carlo (MC), Monte Carlo Tree Search Expectimax (MCTSE) und Temporal Difference (TD) wurden genauer untersucht. Ziel dieser Arbeit ist es, durch die Auswertung des Spiel- und Lernerfolgs, mögliche Stärken und Schwächen der verwendeten Agenten zu erkennen. Insbesondere soll hierbei betrachtet werden, inwiefern sich die besonderen Eigenschaften Blackjacks (rundenbasiertes, nichtdeterministisches Spiel mit imperfekten Informationen) auf die Ergebnisse auswirken.

Um die Spielstärke eines Agenten zu evaluieren, wurde mitunter der durchschnittliche Gewinn des jeweiligen Agenten für verschiedene Parameterkombinationen basierend auf 500 Spielrunden gemessen. In der Messung der Spielstärke konnte MC, dicht gefolgt von MCTSE, die besten Ergebnisse erzielen. Beide Agenten konnten einen durchschnittlichen Gewinn nahe Null erreichen und erzielten somit deutlich bessere Ergebnisse als ein zufällig spielender Agent, welcher einen durchschnittlichen Gewinn von $\approx -4$ erzielt. Für TD-N-Tuple hingegen konnte kein Lernerfolg nachgewiesen werden. Durch Analyse der Spielfunktion von TD-N-Tuple wurde gezeigt, dass es aufgrund der stark fluktuierenden nichtdeterministischen Ereignisse Blackjacks dem TD-N-Tupel Agent nicht möglich ist, Blackjack zu lernen.

Außerdem konnte anhand eines Experiments, dass das Verhalten der Agenten MC und MCTSE am Rundenende untersucht, aufgezeigt werden, dass eine unterschiedlich lange Runde, je nach gewähltem Spielzug (bspw. Hit oder Stand), ein Problem für beide Agenten darstellt. Mithin zeigt sich, dass die untersuchten allgemeinen Agenten bei rundenbasierten, nichtdeterministischen Spielen mit imperfekten Informationen durchaus Limitierungen aufweisen.

# Inhaltsverzeichnis

# 1 Einleitung

Künstliche Intelligenz spielt in immer mehr Lebensbereichen eine wichtige Rolle. So titelt bspw. die Frankfurter Allgemeine Zeitung bereits 2017: “Künstliche Intelligenz überall” (Lindner u. York (2017)). Der Autor führt u.a. als Beispiel Google an. Der Suchriese aus den USA habe bereits alle seine Produkte gemäß dem Prinzip “AI First” überarbeitet. (Vgl. Lindner u. York (2017)) Im spezifischen Bereich Game Learning findet künstliche Intelligenz ebenfalls Anwendung und kann bemerkenswerte Erfolge vorweisen. So besiegte im Jahr 2017 im hochkomplexen Brettspiel Go die künstliche Intelligenz “AlphaGo Zero” den amtierenden Weltmeister (Silver u. Schrittwieser (2017)), was einen Meilenstein des Game Learnings und seines Teilgebiets Reinforcement Learning markiert.

Die Relevanz verbuchter Game Learning und Reinforcement Learning Erfolge der letzten Jahre ist besonders hoch, da für einige Spiele (unter anderem Go) angenommen wurde, dass eine Maschine solch hochkomplexe Spiele niemals meistern wird und dies nur mit menschlicher Intelligenz und Intuition möglich sei. Um alle möglichen Spielpositionen für das Spiel Go zu berechnen ($\approx 2 * 10^{170}$ Spielpositionen (Tromp (2016))), bräuchte es mit heutiger Rechenleistung mehr Zeit als das Universum alt ist. Durch reine Rechenleistung können Spiele wie Go nicht gemeistert werden. Viel mehr wird Mustererkennung, Generalisierung und Abstraktion benötigt, Konzepte, die allgemein hin dem Begriff der Intelligenz zugeordnet werden.

Während für das Spiel Go die extrem hohe Anzahl an möglichen Spielpositionen eine Herausforderung für künstliche Intelligenzen (Agenten) darstellt, können auch andere Klassifizierungen von Spielen, wie zum Beispiel Spiele nichtdeterministischer Natur, sehr komplexe Aufgaben für Agenten darstellen. Bei nichtdeterministischen Spielen besteht die Schwierigkeit, einen Spielverlauf dem Glück bzw. den zufälligen Ereignissen oder den getroffenen Spielentscheidungen zuzuordnen. Somit kann ein Spielzug unabhängig seiner Richtigkeit zum Erfolg führen, jedoch nur, weil der unwahrscheinliche Fall eingetreten ist und vice versa. Eine weitere Klassifizierung von Spielen sind solche imperfekter Informationen, bei denen SpielerInnen keinen Zugriff auf alle im Spiel enthaltenen Informationen besitzen. Für Spiele mit imperfekten Informationen besteht die Herausforderung darin, mit den nicht vollständig verfügbaren Informationen im Spiel umzugehen und die möglichen Ausgänge abzuwägen. Beide Klassifizierungen bieten eine spannende wissenschaftliche Forschungsplattform für künstliche Intelligenzen.

Gegenstand dieser Arbeit ist es, den Spiel- und Lernerfolg von allgemeinen Reinforcement Agenten in einem nichtdeterministischen Spiel imperfekter Informationen in der Game-Learning-Umgebung General Board Game (Konen (2019)) anhand von Blackjack zu untersuchen. Dabei sollen die Agenten Monte Carlo (MC), Monte Carlo Tree Search Expectimax (MCTSE) und Temporal Difference N-Tuple (TD-N-Tuple) genauer untersucht werden. Das stochastische Kartenspiel Blackjack lässt sich des Weiteren als ein rundenbasiertes Spiel mit stark fluktuierenden nichtdeterministischen Ereignis-

sen einstufen. Vorab soll die Blackjack Spielumgebung dem Game-Learning-Framework General Board Game (GBG) unter Einhaltung vorgegebener Interfaces hinzugefügt werden. Um anschließend den Spiel- und Lernerfolg der Agenten zu beobachten und zu vergleichen, sollen Merkmale gefunden werden, nach denen die Spielstärke der allgemeinen Reinforcement Agenten in Blackjack gemessen werden kann. Hauptaugenmerk liegt auf den Herausforderungen beim Lernen und Spielen von Blackjack durch allgemeine Agenten, die sich aus der stark ausgeprägten nichtdeterministischen Natur und den imperfekten Informationen ergeben.

Somit versucht diese Arbeit folgende Forschungsfragen zu beantworten:

i Welche messbaren Spiel- und Lernerfolge können die untersuchten allgemeinen Reinforcement Agenten in Blackjack erzielen?

ii Welchen Einfluss hat die rundenbasierte Natur Blackjacks auf den Spiel- und Lernerfolg der allgemeinen Reinforcement Agent?

iii Inwiefern stellen die stark fluktuierenden nichtdeterministischen Ereignisse Blackjacks ein Problem für die allgemeinen Reinforcement Agenten in Bezug auf ihren Spiel- und Lernerfolg dar?

Ausgehend vom Stand der Forschung werden die Spielregeln von Blackjack erläutert. Infolgedessen wird ein Überblick über das Game-Learning-Framework GBG, der Implementation von Blackjack und der Funktionsweise der untersuchten Reinforcement Agenten gegeben. Zur Beobachtung des Spiel- und Lernerfolgs verwendeter Agenten werden vorab Evaluationsstrategien erarbeitet und anschließend Messungen und Ergebnisse präsentiert. Abschließend werden die gesammelten Messergebnisse sowie aufgetretenen Probleme reflektiert und ein Fazit gezogen.

## 1.1 Stand der Forschung

Im Folgenden werden kurz aktuelle Game Learning Ansätze im Zusammenhang mit Blackjack beleuchtet. Ergebnisse für Blackjack werden meist als durchschnittlich erzielter Payoff angegeben. Der Payoff meint den Gewinn oder Verlust an Spielchips für eine gespielte Runde. Für die meisten Blackjack Varianten nähert sich der durchschnittliche Payoff bei optimaler Strategie null an, bleibt jedoch negativ.

### 1.1.1 RL Card

RL Card ist eine Open Source Game Learning Umgebung[1], die sich auf Kartenspiele spezialisiert. Motivation von RL Card ist, die Herausforderungen anzugehen, welche Reinforcement Learning üblicherweise für Kartenspiele aufweisen. Kartenspiele haben in häufigen Fällen eine sehr hohe Anzahl an möglichen Game States. Auch die Anzahl an möglichen Aktionen ist für Kartenspiele sehr hoch, bei denen das Ausspielen einer Karte eine Aktion darstellt. Außerdem wird abhängig vom Kartenspiel Rivalität oder Kooperation von Agenten gefordert. Eines der in RL Cards implementierten Spiele ist

[1] https://rlcard.org/

eine stark vereinfachte Version von Blackjack. Diese Vereinfachung beschränkt sich auf die Spielzüge Hit und Stand. Für die vereinfachte Version Blackjacks konnte mit einem Q-Learning Agenten ein Lernfortschritt gemessen werden. Der Agent konnte mit steigenden Trainingseinheiten eine Verbesserung im durchschnittlichen Payoff vorweisen, welcher sich einem Wert von -0.05 pro eingesetzten Spielchip annäherte. (Zha u. a. (2019))

### 1.1.2 OpenAI Gym

OpenAI Gym ist ein Opensource Reinforcement Learning Projekt,[2] welches die Philosophie verfolgt nur Spielumgebungen zu abstrahieren, nicht aber die Agenten. Somit erhalten Entwickler mehr Freiraum beim Design ihrer Agenten. Dies soll einen regen Austausch und Vergleich von Agenten zwischen Forschenden anregen. Durch die Standardisierung der Spielumgebung versucht OpenAI Gym indirekt die fehlende Standardisierung von Publikationen im Bereich des Game Learnings zu verbessern. Die fehlende Standardisierung wird benötigt, da kleine Unterschiede im Design der Rewards (Belohnung für Agenten) oder den Actions (Möglichkeit mit der (Spiel-)Umgebung zu interagieren) einen großen Unterschied in der Schwierigkeit der Aufgabe darstellen können. OpenAI Gym stellt eine vereinfachte Spielumgebung für Blackjack bereit. Die Spielzüge wurden wie auch von RL Card auf Hit und Stand beschränkt. (Brockman u. a. (2016))

### 1.1.3 Computer Science Project in Blackjack

Der Computer Science Projekt Report von Dylan Clark, Lana Gorlinski und Nathan Ondracek [3] behandelt eine Version von Blackjack, welche alle regulären Spielzüge implementiert. In ihrer Arbeit wird der Expectimax sowie der Q-learning Agent eingesetzt. Der Expectimax Agent erreichte nach $10^6$ Spielrunden einen durchschnittlichen Payoff von -0.15 pro eingesetzten Spielchip. Der Q-Learning Agent erzielte nach $10^6$ Trainingsrunden und anschließend $10^6$ Evaluationsrunden einen durchschnittlichen Payoff von -0.002 pro eingesetzten Spielchip und trifft dabei in über 90% der Fälle eine Entscheidung entsprechend der optimalen Strategie.

### 1.1.4 Einsatz eines Q-Learning Agent in Blackjack

Charles de Granville untersucht in seiner Publikation den Lernerfolg eines Q-Learning Agent in Blackjack. Seine implementierte Version Blackjacks setzt abgesehen von Surrender alle regulären Spielzüge um. Der Q-Learning Agent erreicht nach $10^7$ Trainingsrunden einen durchschnittlicher Payoff im Berreich von $-0.24$ bis $-0.1$ und nähert sich somit dem durchschnittlichen Payoff der optimalen Strategie an. Im Fazit resümiert Charles de Granville: „These results are encouraging, but there remains room for improvement. A better exploration strategy, such as the Bayesian Q-learning algorithm, may lead to an improvement in the results.“ (De Granville, S. 3)

[2] https://gym.openai.com
[3] https://static1.squarespace.com/static/5a346e09cf81e0cf09ea581b/t/5c6a6355b208fc289898291f/1550476118802/cs182+final+report.pdf

# 2 Blackjack

Blackjack ist ein weit verbreitetes Kartenspiel mit stochastischen Elementen. Jede Karte in Blackjack repräsentiert einen numerischen Wert. Die Spielbeteiligten versuchen, einen höheren Handwert als der Dealer zu erreichen, ohne dabei die Summe von 21 zu überschreiten. Trotz des Nichtdeterminismus können in Blackjack Strategien verwendet werden, welche die höchsten Gewinnchancen für eine spezifische Situation bringen. Bei mathematisch perfekter Strategie verlieren Spielbeteiligte durchschnittlich weniger als 1% des Einsatzes. (Baldwin u. a., 1956, S. 439) Blackjack ist ein Spiel mit imperfekten Informationen, da nicht zu jedem Zeitpunkt alle sich im Spiel befindenden Karten offengelegt sind.

## 2.1 Regeln

Blackjack kann von bis zu sieben Spielern unabhängig voneinander gegen den Dealer gespielt werden. Am weitesten verbreitet ist das Spiel mit sechs Decks französischer Spielkarten, welche jeweils aus 52 Karten bestehen.

### 2.1.1 Kartenwerte

Jede Karte repräsentiert einen Zahlenwert.

- Ein Ass kann eins oder elf zählen. Es wird immer zugunsten des Spielers, welcher das Ass besitzt, gewertet. Die Hand bestehend aus Ass und neun bildet den numerischen Wert 20, das Ass zählt in diesem Fall elf. Die Hand bestehend aus Ass, neun und fünf bildet den numerischen Wert 15, das Ass zählt in diesem Fall eins, da sonst der Schwellenwert von 21 überschritten würde. Eine Hand, bei der das Ass elf zählt, wird soft genannt.
- Buben, Damen und Könige zählen zehn.
- Alle weiteren Karten zählen entsprechend ihrer Zahl.

### 2.1.2 Gewinnbedingung

Ein Spieler gewinnt gegen den Dealer, wenn der Wert seiner Hand (die Summe seiner Kartenwerte) größer als die des Dealers ist. Überschreitet der Handwert des Spielers die Zahl 21, verliert er diese, unabhängig von den Karten des Dealers. Überschreitet der Handwert des Dealers die Zahl 21, gewinnt jeder Spieler mit einem Handwert kleiner als 22. Haben Spieler und Dealer den gleichen Handwert, gilt ein Unentschieden (auch Push genannt).

### 2.1.3 (Natural) Blackjack

Erhält ein Spieler mit seinen ersten zwei Karten einen Handwert von 21, so besitzt er einen Blackjack (im Folgenden auch Natural genannt) und gewinnt gegen jede andere Hand. Ein Beispiel hierfür sei die Hand bestehend aus einem Ass und einem König. Hat sowohl der Spieler als auch der Dealer einen Natural, so wird dies als ein Unentschieden gewertet. Ein Natural kann nicht in einer geteilten Hand auftreten bzw. wird nicht als Natural gewertet.

### 2.1.4 Spielzüge

**Bet** Der Spieler platziert den gewünschten Spieleinsatz vor dem Erhalt seiner Handkarten und gibt den Zug an den nächsten Spieler ab. Im Verlauf dieser Arbeit ist der Spielzug Bet auf einen festen Spieleinsatz von zehn fixiert.

**Hit** Der Spieler fordert eine weitere Karte vom Dealer.

**Double Down** Nur erlaubt mit exakt zwei Karten in der Hand. Der Spieler darf seinen Einsatz der Hand verdoppeln, erhält noch exakt eine Karte und gibt seinen Zug an den nächsten Spieler ab.

**Split** Nur erlaubt mit exakt zwei Karten desselben Ranges in der Hand. Der Spieler muss den Einsatz für die zu teilende Hand erneut erbringen. Die Hand wird geteilt und beide Hände werden anschließend unabhängig voneinander gespielt. Erhält ein Spieler in einer geteilten Hand einen Handwert von 21 mit exakt zwei Karten, gilt diese nicht als Natural. Ein Beispiel für eine geteilte Hand sei die Hand bestehend aus zwei Buben, welche nach Split zwei neue Hände mit jeweils einem Buben ergeben. Die Hand, auf welche das Teilen angewendet wurde, wird zuerst gespielt und bekommt eine zweite Karte ausgeteilt. Die Anzahl an Händen für Spieler ist auf drei begrenzt.

**Surrender** Muss der erste Spielzug der Hand sein. Der Spieler gibt seine Hand auf und erhält die Hälfte seines Einsatzes der Hand zurück, verliert jedoch jegliche Gewinnchancen. Nach einem Split darf ebenfalls der Spielzug Surrender nicht mehr gewählt werden.

### 2.1.5 Spielablauf

Vor dem Erhalt der Karten platziert jeder Spieler eine Anzahl von Spielchips, welche er für die auszuteilende Hand setzen möchte. Nach Platzierung der Einsätze teilt der Dealer im Uhrzeigersinn den Spielern und sich selbst zwei Karten aus. Die Karten der Spieler werden offen ausgeteilt und vor den Spielern auf dem Tisch platziert. Die erste Karte des Dealers wird ebenfalls offen ausgeteilt, die zweite Karte bleibt jedoch verdeckt. Falls es möglich ist, mit der offenen Karte des Dealers einen Natural zu bilden, prüft der Dealer seine Hand auf einen Natural, indem er sich seine verdeckte Karte anschaut. Besitzt der Dealer einen Natural, ist die Runde sofort beendet, andernfalls schreitet die Runde voran. Ist die offene Karte des Dealers ein Ass, haben die Spieler

die Möglichkeit sich durch Abschließen einer Insurance gegen einen Natural des Dealers zu versichern. Die Spieler wählen nun, angefangen links vom Dealer, im Uhrzeigersinn Spielzüge, bis jeder Spieler seinen Zug abgegeben hat und der Dealer am Zug ist. Der Dealer folgt unabhängig von den Händen der Spieler festen Vorgaben bei der Wahl seiner Spielzüge. Erst wird die verdeckte Karte des Dealers aufgedeckt: Besitzt der Dealer einen Handwert kleiner 17, wählt er den Spielzug Hit, andernfalls wählt er den Spielzug Stand. Hat der Dealer seine Hand beendet, wird ermittelt, welche Hände der Spieler gegen den Dealer gewonnen, verloren oder ein Unentschieden erzielt haben, danach wird dementsprechend ausgezahlt. Die Runde ist beendet. Alle verwendeten Karten werden beiseite gelegt und eine neue Runde beginnt. (Meißner (2021))

### 2.1.6 Auszahlungen

Verliert die Hand eines Spielers gegen die Hand des Dealers, ist der zugehörige Einsatz verloren. Bei einem Unentschieden gegen den Dealer erhält der Spieler seinen Einsatz zurück. Gewinnt ein Spieler gegen den Dealer, wird im Verhältnis 1 zu 1 ausgezahlt. Gewinnt ein Spieler mit einem Natural gegen den Dealer, wird im Verhältnis 3 zu 2 ausgezahlt. Beispiel für eine Auszahlung: Ein Spieler, welcher mit einem Natural gewonnen hat und 100 Spielchips eingesetzt hat, wird im Verhältnis 3 zu 2 ausgezahlt. Der Spieler erhält seinen Einsatz in Höhe von 100 Spielchips zurück und bekommt einen Gewinn in Höhe von $100 * \frac{3}{2} = 150$ Spielchips ausgezahlt.

### 2.1.7 Insurance Nebenwette

Nach dem Austeilen der Karten an die Spielbeteiligten und den Dealer prüft der Dealer, ob er einen Natural hat. Das Prüfen auf einen Natural findet statt, bevor Spielende Spielzüge für ihre Hand machen können. Hat der Dealer einen Natural, ist die Spielrunde beendet, bevor die Spielenden Spielzüge für ihre Hand machen, die Auszahlung erfolgt jedoch weiterhin. Hat der Dealer keinen Natural, wird die Runde fortgesetzt. Die Nebenwette Insurance kann abgeschlossen werden, wenn die offene Karte des Dealers ein Ass zeigt und gilt als gewonnen, wenn der Dealer einen Natural besitzt. Die Einsatzhöhe der Insurance Nebenwette ist ebenfalls auf zehn Spielchips fixiert. Die Besonderheit der Nebenwette Insurance ist, dass sie vor dem Prüfen auf einen Natural des Dealers abgeschlossen wird. Sie bietet somit den Spielbeteiligten die Möglichkeit mit dem Spiel zu interagieren, auch wenn der Dealer einen Natural hält. Dies wäre ohne die Nebenwette nicht möglich. Bei der gewonnenen Nebenwette Insurance wird der Wetteinsatz zurückerstattet und ein Gewinn im Verhältnis 2 zu 1 ausgezahlt.

**Gewinnchancen** Besteht für die Spielenden die Möglichkeit zum Abschluss der Nebenwette Insurance, zeigt der Dealer ein Ass. Um einen Natural zu erlangen, benötigt der Dealer eine Zehn, Bube, Dame oder König. Bei 13 möglichen Karten, die aufgedeckt werden können, führen 4 dieser Karten zu einem Natural, also einer Wahrscheinlichkeit von $\frac{4}{13}$, dass der Dealer einen Natural besitzt und eine Wahrscheinlichkeit von $\frac{9}{13}$, dass der Dealer keinen Natural besitzt.

Mit diesen Informationen lässt sich der Erwartungswert des Gewinns (G) dieser Nebenwette berechnen:

$$E(G) = \frac{4}{13} * 2 - \frac{9}{13} * 1 = \frac{-1}{13} = -0.0769 \tag{2.1}$$

Für jeden Spielchip, der für die Nebenwette Insurance eingesetzt wird, wird ein Verlust von 0.079 Spielchips erwartet.

## 2.2 Strategien

### 2.2.1 Basic Strategy

Die Basic Strategy gibt den besten Spielzug für eine beliebige Spielsituation (im Kontext des Game Learnings ein beliebiger Game State) an. Es werden lediglich Informationen verwendet, welche im Moment der Betrachtung der Spielsituation ersichtlich sind. Aus den gegebenen Informationen wird die Gewinnwahrscheinlichkeit für jeden Spielzug in der spezifischen Situation berechnet. Der Spielzug, der im Mittel zum höchsten Gewinn führt, wird als bester Spielzug angegeben. Die Basic Strategy wird kompakt als Tabelle bereitgestellt. So kann aus der Tabelle der beste Spielzug abgelesen werden, ohne Gewinnwahrscheinlichkeiten berechnen zu müssen. Für verschiedene Varianten von Blackjack existieren korrespondierende Basic Strategy Tabellen. Jede mögliche Spielsituation hat die gleiche Eintrittswahrscheinlichkeit, solange die noch im Deck befindlichen Karten unbekannt sind. Dementsprechend kann nicht vorausgesagt werden, ob eine vorteilhafte oder nachteilige Spielsituation eintreten wird. Folglich gibt die Basic Strategy keine Angaben über die Höhe des Wetteinsatzes an. Aufgrund des negativen Erwartungswerts des Gewinns beim Abschluss der Nebenwette Insurance wird jene nach der Basic Strategy niemals abgeschlossen.

### 2.2.2 Kartenzählen

Kartenzählen ist eine Strategie für Blackjack, welche über die Basic Strategy hinausgeht. Durch Beobachtung der ausgeteilten Karten kann auf den ungefähren Zustand des Decks geschlossen werden. Mit diesen zusätzlichen Informationen kann von der Basic Strategy abgewichen werden. Für das Kartenzählen wurden verschiedene Systeme entwickelt. Das einfachste ist das Hi-Lo System. In diesem wird den Karten der Wert -1, 0, oder 1 zugewiesen. (zwei bis sechs = 1 — sieben bis neun = 0 — zehn, Bube, Dame, König, Ass = -1). Nun kann bei jeder ausgeteilten Karte ihr zugewiesener Wert auf den sogenannten Count summiert werden. Hat der Count einen positiven Wert, sind mehr hohe als niedrige Karten im Deck und vice versa. Statistisch gibt ein Deck mit vielen hohen Karten den Spielenden einen Vorteil, während ein Deck mit vielen niedrigen Karten für den Dealer von Vorteil ist. Folglich können mit diesen zusätzlichen Informationen Entscheidungen getroffen werden, welche sich auf die Höhe des Wetteinsatzes beziehen. Auch die Nebenwette Insurance wird ab einem ausreichend hohen Count als vorteilhaft bewertet. Da für weiterführende Konzepte wie die des Kartenzählens die Voraussetzung ein endliches Deck ist und in der Implementation von Blackjack ein unendliches Deck verwendet wird, werden weiterführende Konzepte in dieser Arbeit nicht weiter betrachtet. [1]

[1] https://wizardofodds.com/games/blackjack/card-counting/high-low/

# 3 General Board Game (GBG)

General Board Game (GBG) wird von Hauptentwickler Wolfgang Konen als eine Game Learning Umgebung mit verschiedenen Agenten bereitgestellt. Motivation des GBG ist es, übliche Prozeduren im Bereich des Game Learnings zu generalisieren und damit Agenten beliebige zugbasierte Spiele lernen und spielen zu lassen. Hierdurch wird ein einfacher Einstieg für Studierende in den Bereich des Game Learnings ermöglicht, da das Spiel sowie die Agenten samt ihrer Logik nicht mehr entwickelt werden müssen, sondern lediglich das Spiel nach Vorgaben des GBG Interfaces implementiert werden muss. Durch Einhaltung der Interfaces ist gewährleistet, dass jedes implementierte Spiel die benötigte Funktionalität besitzt, um den Agenten das Lernen und Spielen zu ermöglichen. Infolgedessen wird den Studierenden die Beantwortung von Forschungsfragen ermöglicht, die ansonsten durch den großen Programmier-Overhead nicht möglich wäre. (Konen (2019))

Durch das große Interesse der Studierenden am Game Learning entstehen so regelmäßig Projekte innerhalb des GBG, welche zu Iterationen bzw. weiterer Generalisierung des Frameworks anregen. [1]

## 3.1 Überblick

Im Folgenden wird ein kurzer Überblick über das GBG Framework gegeben. Die nachfolgenden Informationen sind weitgehend aus dem dieser Bachelorarbeit vorgelagertem Praxisprojekt (Meißner (2021)) entnommen.

### 3.1.1 Game State

Ein Game State ist der Spielzustand zu einem beliebigen Zeitpunkt. Eine chronologische Aneinanderreihung von Game States bildet einen Spielablauf. Es müssen solche Informationen im Game State bereitgestellt werden, aus welchen sich auf alle anderen, nicht gespeicherten, jedoch spielrelevanten Informationen schließen lässt.

### 3.1.2 Interface StateObservation

Ein State Observer implementiert das Interface StateObservation und koordiniert den gesamten Spielablauf. Die Aufgaben eines State Observers des GBG sind unter anderem die Repräsentation des Game States, den Game State auf Basis eines Spielzuges (Action) in den Folgezustand zu überführen, legale Actions des aktuellen Spielers zu ermitteln und Belohnungen an Agenten auszuschütten (Rewards). Ein State Observer überführt den Game State durch den Aufruf der Funktion advance(Action) in den Folgezustand (Advance).

[1] https://github.com/WolfgangKonen/GBG

### 3.1.3 Interface StateObsNonDeteterministic

Das Interface StateObsNonDeterministic erweitert das Interface StateObservation und erfordert zusätzliche Funktionalitäten des State Observers, welche für nichtdeterministische Spiele benötigt werden. Das Advance eines nichtdeterministischen State Observers wird in einen deterministischen und einen nichtdeterministischen Teil aufgeteilt.

### 3.1.4 Interface GameBoard

Das Game Board eines Spiels implementiert das GameBoard Interface und realisiert somit die Schnittstelle zwischen Mensch und Spiel. Möchte ein Mensch einen Zug ausführen, wird dieser vom Game Board entgegengenommen und ein Advance des spielspezifischen State Observers initiiert. Des Weiteren besitzt und verwaltet das Game Board eine Referenz zur grafischen Oberfläche und aktualisiert diese nach einem Advance des State Observers.

### 3.1.5 Abstrakte Klasse Arena

Die Arena eines Spiels wird von der Basisklasse Arena abgeleitet. Die Arena eines Spiels erlaubt es, das Spiel mit gewünschten Voreinstellungen zu starten. Hier kann gewählt werden, welche Agenten (oder Menschen) die freien Plätze der Spieler einnehmen und gegeneinander antreten sollen.

### 3.1.6 Interface PlayAgent

Das Agent Interface erlaubt es, das GBG um noch nicht vorhandene Agenten zu erweitern. Jedoch stehen bereits viele Agenten zur Verfügung, welche bereit sind beliebige zugbasierte Spiele nach ihrer Strategie zu erlernen oder direkt zu spielen. Für einen tieferen Einblick in die zur Verfügung stehenden Agenten und Interfaces empfiehlt sich der technische Bericht zum GBG Framework (Konen, 2020, S. 11).

### 3.1.7 Evaluator

Der Evaluator eines Spiels implementiert das Evaluator Interface und dient dazu die Spielstärke verwendeter Agenten zu evaluieren. Die Evaluationsstrategien zur Bewertung der Spielstärke müssen spielspezifisch implementiert werden. Nach Implementation eines Evaluators können beispielsweise beim Trainingsprozess der Agenten Evaluationsergebnisse grafisch dargestellt werden.

## 3.2 Blackjack im GBG

Ziel der Umsetzung von Blackjack war es Blackjack möglichst realitätsgetreu zu implementieren. Aus diesem Grund wurden alle in den Regeln aufgezählten Spielzüge und auch die Nebenwette Insurance implementiert. Außerdem wurde eine grafische Oberfläche realisiert um Menschen die Möglichkeit zu geben, mit der Blackjack Umgebung und den Agenten zu interagieren. Eine aktuelle Version der grafischen Blackjack Umgebung ist auf Abbildungen 3.1 und 3.2 abgebildet.

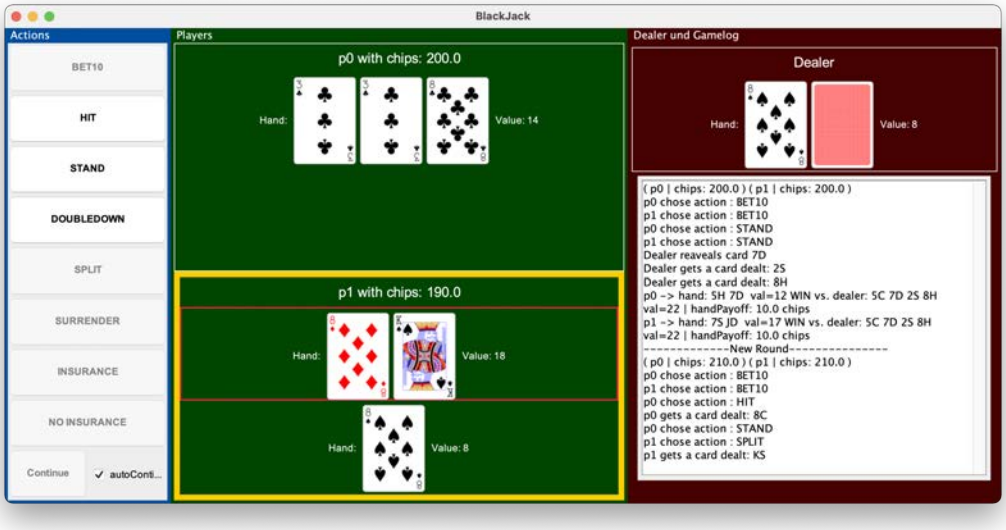


Abbildung 3.1: Grafische Oberfläche Blackjack (Spieler „p1“ mit einer geteilten Hand am Zug)


Abbildung 3.2: Grafische Oberfläche Blackjack (Rundenende resultierend aus Abbildung 3.1)

## 3.3 Klassenüberblick

Das Klassendiagramm 3.3 visualisiert die untereinander auftretenden Zusammenhänge der zur Umsetzung von Blackjack verwendeten Klassen. Eine graue Raute stellt eine Aggregation dar. Eine schwarze Raute bildet eine Komposition.

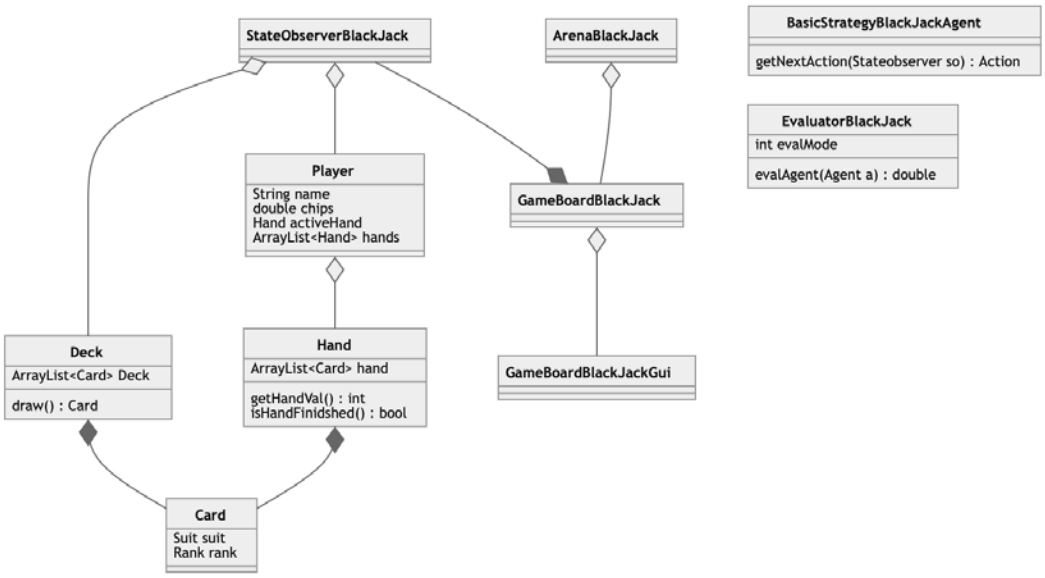


Abbildung 3.3: Klassendiagramm der Blackjack Umgebung

**Die Klasse StateObserverBlackJack** implementiert das Interface StateObsNonDeterministic. Stellt einen Game State in Blackjack dar und bietet Funktionalitäten mit diesem zu interagieren.

**Die Klasse Player** kodiert spielrelevante Informationen eines Spielbeteiligten und bietet Funktionen um diese zu manipulieren.

**Die Klasse Hand** kodiert eine Hand in Blackjack und bietet Funktionen um diese zu manipulieren. Eine Hand besteht aus einer oder mehreren Karten.

**Die Klasse Deck** kodiert das unendliche Deck in Blackjack und bietet Funktion eine Karte zu ziehen. Da keine spezifische Instanz eines unendlichen Decks benötigt wird, wurde die Klasse Deck als Statisches Objekt in der Klasse ArenaBlackJack bereitgestellt. Dies beschleunigt den Kopierprozess eines Game States der Klasse StateObserverBlackJack.

**Die Klasse BasicStrategyBlackJackAgent** implementiert das Interface PlayAgent und setzt einen für Blackjack spezifischen Agenten um.

**Die Klasse EvaluatorBlackJack** implementiert das Interface Evaluator und setzt Evaluationsstrategien für Blackjack um.

**GameBoardBlackJackGui** realisiert die grafische Oberfläche der Blackjack Umgebung.

## 3.4 Besonderheiten der Blackjack Implementierung im GBG

### 3.4.1 Partial State

Blackjack ist neben Poker eines der ersten Spiele mit imperfekten Informationen in GBG. Aus diesem Grund wurde das Konzept eines Partial States umgesetzt. In einem Partial State werden diejenigen Informationen unkenntlich gemacht, die ein Agent nicht kennen darf. Ein Partial State vervollständigt sich zufällig, sobald die unkenntlich gemachten Informationen benötigt werden. Somit können Agenten ohne Weiteres eine Simulation auf einem Partial State ausführen. Die benötigten unkenntlichen Informationen werden im Laufe einer Simulation des Partial State bereitgestellt, können sich von den Informationen des Master Game State unterscheiden. Für Blackjack herrscht hier jedoch eine potenzielle Fehlerquelle. Eine zufällige Vervollständigung eines Partial State kann in zwei Situationen auftreten:
**1)** Der Dealer prüft seine Hand auf einen Natural.
**2)** Der Dealer spielt seine Hand bzw. deckt seine verdeckte Karte auf.
Besitzt der Dealer in Situation **1)** beim Prüfen auf einen Natural tatsächlich einen Natural, ist die Runde umgehend beendet. Folglich darf beim Vervollständigen von Situation **2)** dem Dealer nicht zufällig ein Natural vervollständigt werden. Hätte der Dealer einen Natural, wäre die Runde schon nach dem Prüfen auf einen Natural (Situation **1)**) beendet worden. Diese Unterscheidung spielt besonders in den Fällen eine Rolle, in denen ein spezifischer Game State konstruiert wird (z.B. beim Evaluieren der Agenten), welcher sich inmitten der Runde befindet. In diesen Situationen hat die Prüfung auf einen Natural bereits stattgefunden.

### 3.4.2 Rundenende

Blackjack ist neben Poker eines der ersten Spiele im GBG, in welchem der Spielablauf eine Aneinanderreihung von Runden ist. Eine bestimmte Anzahl von Runden stellt eine Episode dar. Ist eine Runde beendet, beginnt eine neue Runde, die nur marginal von der vorangegangenen Runde beeinflusst wird. Gegebenenfalls wurde durch die alte Runde der Chipstand der Spieler verändert. Ist jedoch der Chipstand (wie in Blackjack) nicht von Bedeutung für Spielstrategien, können die Runden unabhängig voneinander betrachtet werden. Des Weiteren fehlte dem GBG die Funktionalität den Spielablauf an einem gewünschten Zeitpunkt zu stoppen (dem Rundenende) und den Game State zu diesem Zeitpunkt darzustellen. Aus diesem Grund wurden die Interfaces des GBG Frameworks, dahingehend angepasst, dass State Observer die Möglichkeit haben, ein Rundenende zu markieren, wenn dieses erreicht wurde. Diese Änderung wurde nicht nur mit der Intention erarbeitet, den Spielablauf am Rundenende zu stoppen und

darzustellen, sondern auch um den Agenten die Möglichkeit zu geben ihre Simulationen nur bis zum Erreichen des Rundenendes durchzuführen.

### 3.4.3 Game-Score und Reward

Ein Reward im GBG dient als Belohnung für einen Agenten. Der Game-Score gibt den Spielpunktestand des Agenten an. Für viele Spiele im GBG ist der Game-Score gleich dem Reward. Der Game-Score gibt in Blackjack den aktuellen Chipstand eines Agenten an und beschreibt somit den konkreten Punktestand des Game States. Der Reward ist der Chipstand des Agenten am Ende einer Runde und beschreibt den Erfolg der gespielten Runde. Der Reward darf erst am Ende einer Runde eine Veränderung annehmen, denn erst am Ende der Runde steht fest, ob der Agent die eingesetzten Chips verliert, zurückerhält, oder einen Gewinn erzielt. Der Game-Score verändert sich sofort beim Einsetzen der Chips. Der Reward wird als absolute Zahl angegeben und repräsentiert die Chips, die ein Agent am Rundenende besitzt. Die Veränderung des Rewards kann sich in Blackjack bei einer fixierten Einsatzstufe von 10 Spielchips von -60 bis +60 erstrecken. Der Reward nimmt diese Veränderung erst an einem Rundenende an, in allen anderen Game States ist die Veränderung des Rewards 0.

## 3.5 Agenten

Im Laufe der Arbeit werden die Reinforcement Learning Agenten Monte Carlo, Monte Carlo Tree Search Expectimax und und Temporal Difference N-Tuple näher in Zusammenhang mit Blackjack betrachtet. Eine detaillierte Übersicht über die im GBG vorhanden Agenten ist im technischen Report von Wolfgang Konen zu finden. [2]

### 3.5.1 Reinforcement Learning

Beim Reinforcement Learning dient die Umgebung und ihre Antwort auf das Verhalten des Agenten gleichzeitig als Input für den Agenten. Der Agent kann über Actions mit

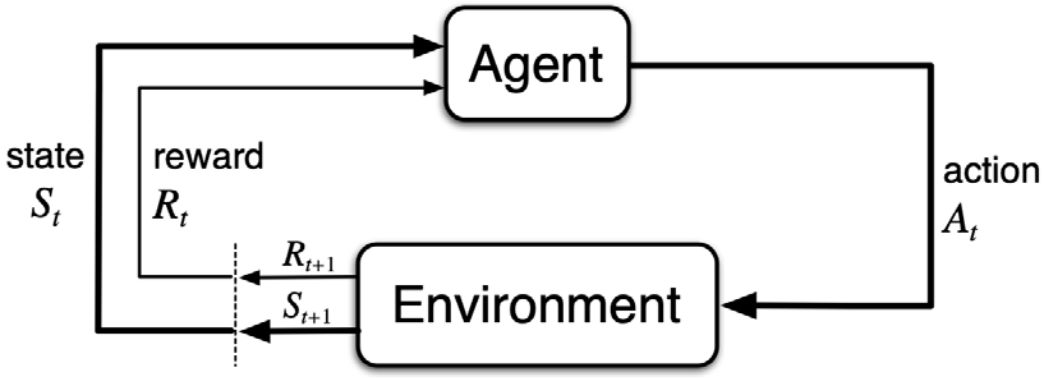


Abbildung 3.4: Reinforcement Learning (Sutton u. Barto, 2018, S. 48)

[2] http://www.gm.fh-koeln.de/ciopwebpub/Konen20d.d/TR-GBG.pdf

der Umgebung interagieren, welche abhängig von der gewählten Action ihren Zustand verändert und einen positiven, neutralen oder negativen Reward ausschüttet. Abhängig von den ausgeschütteten Rewards können Agenten ihr Verhalten anpassen und somit aus ihrem Verhalten lernen.

> „Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging cases, actions may affect not only the immediate reward but also the next situation and, through that, all subsequent rewards. These two characteristics—trial-and-error search and delayed reward—are the two most important distinguishing features of reinforcement learning.“ (Sutton u. Barto, 2018, S. 1)

Reinforcement Learning Agenten unterscheiden sich untereinander darin, mit welcher Strategie sie versuchen ihren Reward (wie von Sutton und Barto beschrieben) zu maximieren.

### 3.5.2 Monte Carlo (MC)

Auf den Monte Carlo Methoden basierende Agenten wurden erstmals von Abramson(1990) für die Spiele Schach, TicTacToe und Othello verwendet. Weitere (auch stochastische) Spiele folgten gegen Ende der Neunzigerjahre. (Chaslot, 2010, S. 16) Die Monte Carlo Methode wurde jedoch bereits früher in der Physik eingesetzt:

> „The Monte Carlo method was given its name by Stanislaw Ulam and John von Neumann, who invented the method to solve neutron diffusion problems at Los Alamos in the mid 1940s.“ (Shonkwiler u. Mendivil, 2009, S. 1)

Die Idee einer Monte Carlo Simulation ist es, durch Wiederholung eines Zufallsexperiments eine Stichprobe aus unabhängig und identisch verteilten Zufallsvariablen zu erhalten, um die Kenngrößen der gewünschten Zielverteilung (meistens sehr komplexe Dichten) zu approximieren. (Andrieu u. a. (2003)) Je höher der Umfang der (simulierten) Stichprobe ist, desto kleiner wird der Fehler der Approximation. Ein sehr einfaches Beispiel für eine solche Simulation könnte ein Münzwurf sein, bei dem nicht bekannt ist, ob die Münze fair ist oder nicht. Nach einem einzigen Wurf, bei dem Kopf geworfen wurde, lässt sich schlecht auf die Wahrscheinlichkeitsverteilung von Kopf und Zahl schließen. Nach 1000 Würfen kann jedoch eine deutlich bessere Prognose der Wahrscheinlichkeitsverteilung von Kopf und Zahl gegeben werden.

Erste Ideen dieser Art der Problemlösung gehen zurück bis in das Jahr 1733, in dem Comte de Buffon das Gedankenexperiment zum Nadelproblem durchführte, um die Kreiszahl Pi zu approximieren. (Aigner u. Ziegler (2010))

#### 3.5.2.1 Funktionsweise

Die Funktionsweise des Monte Carlo Agenten (MC Agent) in GBG lässt sich wie folgt zusammenfassen: Für jeden Spielzug bzw. jede Action $a$, die im aktuellen Game-State

$s$ verfügbar ist, wird $s$ kopiert und mit der Action $a$ in den Folgezustand $s'$ überführt. Für die neu entstandenen Game-States $s'$ wird eine zufällige Simulation (Rollout) gestartet, welche zufällige Actions auswählt und anwendet, bis ein Spielende oder eine gewünschte Tiefe erreicht wurde. Der Reward in diesem zufällig entstandenen Game-State wird der ersten korrespondierenden Action $a$ aus $s$ zugeordnet. Dieser Prozess wird unter Beachtung der Rechenzeit so oft wie möglich wiederholt, um anschließend aus den gesammelten Rewards für jede Action $a$ in $s$ einen repräsentativen Durchschnitt zu bilden. Die gewünschte Simulationstiefe(Rolloutdepth) und die Anzahl an Wiederholungen lassen sich in GBG durch die Parameter Rolloutdepth und Iterations bestimmen. Durch die stochastische Natur des MC Agent wird erwartet, dass der Agent gute Resultate in Blackjack erzielen kann.

### 3.5.3 Monte Carlo Tree Search Expextimax (MCTSE)

Der Monte Carlo Tree Search Expectimax Agent (MCTSE) basiert auf dem Monte Carlo Tree Search Agent (MCTS) und wurde für nichtdeterministische Spiele entwickelt. MCTSE versucht das Problem von MCTS zu lösen, bei welchem im Suchbaum von MCTS zufällige Ereignisse nicht dargestellt werden und somit nicht über diese zufälligen Ereignisse gemittelt werden kann. Die fehlende Darstellung von zufälligen Ereignissen in MCTS stellt ein Problem für nichtdeterministische Spiele dar. Zur Lösung dieses Problems führt MCTSE eine neue Knotenart, den Chance Knoten, ein. Auf eine Action, dargestellt in einem Expextimax Knoten, folgen eine beliebige Anzahl von Chance Knoten, welche die möglichen Resultate des nichtdeterministischen Ereignisses darstellen. Der MCTSE Agent wurde von Kutsch(2017) in GBG für das nichtdeterministische Spiel 2048 implementiert.

#### 3.5.3.1 Funktionsweise

MCTSE durchläuft wie auch MCTS die Phasen Selection, Expansion, Simulation und Backpropagation.

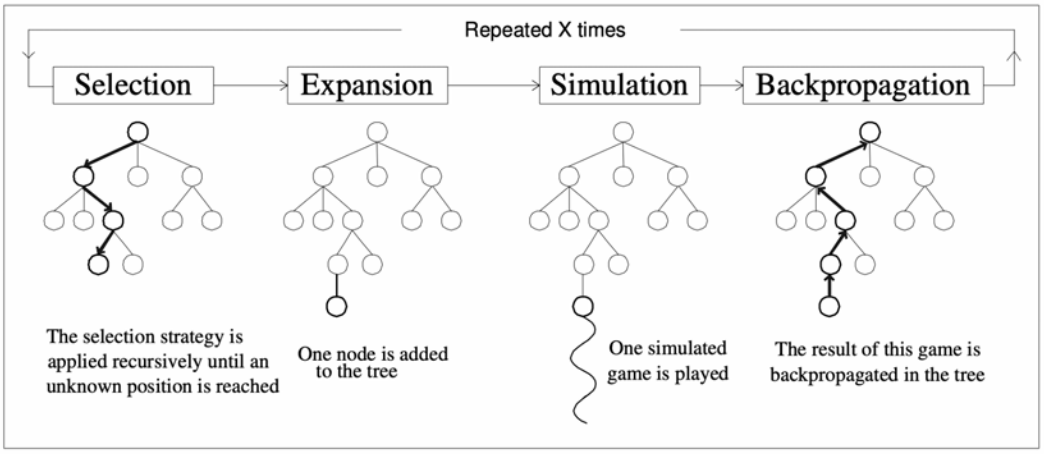


Abbildung 3.5: MCTSEPhasen (Chaslot, 2010, S. 18)

**Selection** In der Selection Phase wird der Baum des Agenten nach einer bestimmten Selektionsstrategie traversiert, bis ein unbekannter Knoten erreicht wurde. Die Selektionsstrategie bestimmt eine Balance zwischen der Auswahl der Knoten, welche bisher das beste Resultat lieferten (Exploitation) und noch unbekannten Knoten (Exploration). (Chaslot, 2010, S. 19) (Świechowski u. a., 2021, S. 7) Für MCTSE gilt es jedoch zwischen Expectimax Knoten und Chance Knoten zu unterscheiden. Denn ein Chance Knoten repräsentiert einen möglichen Game State, wohingegen ein Expectimax Knoten eine mögliche Action in diesem Game State darstellt. Im Gegensatz zum von Chaslot (2010) behandelten MCTS Agent, bei dem nur eine Knotenart existiert, wird beim MCTSE Agent ein Chance Knoten selektiert. Wurden für den aktuellen Chance

Knoten noch nicht alle Actions expandiert, bleibt der Knoten gewählt und es folgt die Expansion Phase. Sind für den gewählten Chance Knoten bereits alle Actions als Expectimax Knoten angelegt, wird nach der Selektionsstrategie einer dieser Expectimax Knoten ausgewählt, die korrespondierende Action auf den Game State angewendet und das Resultat als Chance Knoten dem ausgewählten Expectimax Knoten angehangen. Der neu erstellte Chance Knoten wird anschließend ausgewählt. Es folgt die Expansions Phase. (Kutsch, 2017, S. 40) In Abbildung 3.6 ist ein sehr einfacher MCTSE Baum für Blackjack abgebildet. Hierbei soll die Hand des Dealers und die Hand des Spielbeteiligten den Game State repräsentieren. An jede Action (Expectimax-Knoten)

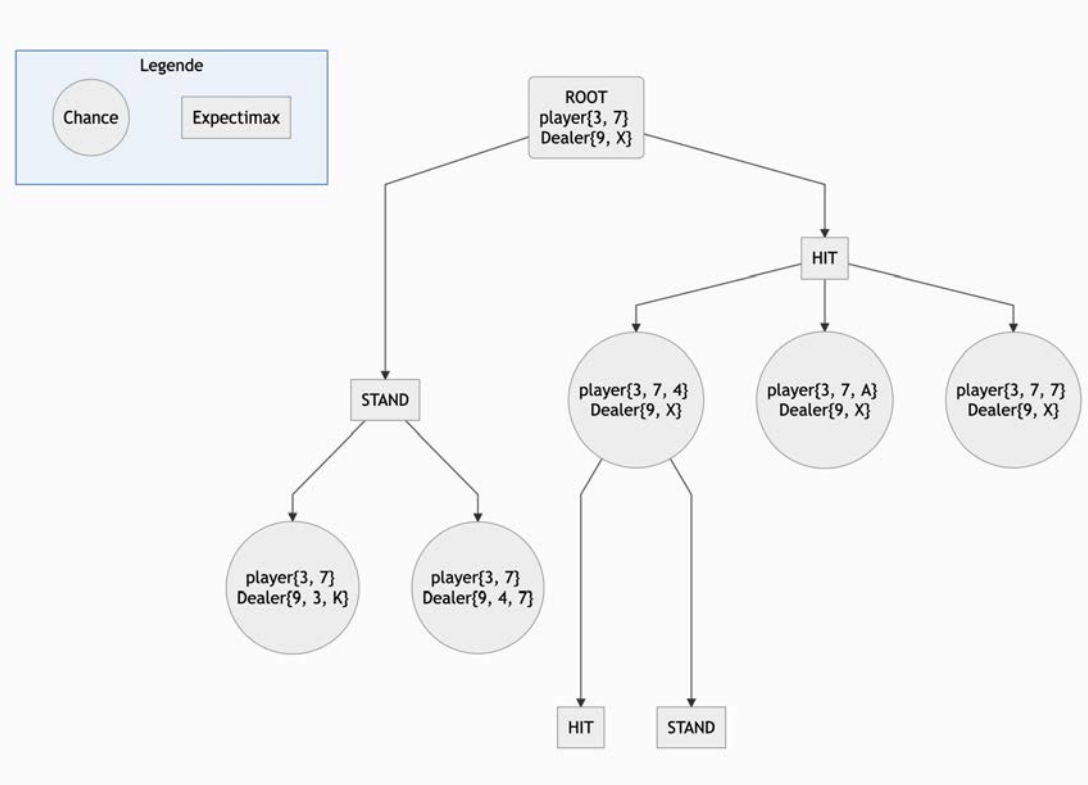


Abbildung 3.6: MCTSE einfacher Blackjack Baum

können also einige Chance Knoten angehängt werden.

**Expansion** Wurde durch die Selection Phase ein Chance Knoten ausgewählt, wird für diesen (falls noch nicht die maximale Baumtiefe erreicht wurde) eine noch nicht expandierte Action ausgewählt. Für diese Action wird ein Expectimax Knoten angelegt, die gewählte Action auf den Game State angewendet und das Resultat als Chance Knoten dem erstellten Expectimax Knoten angehangen. (Kutsch, 2017, S. 43)

**Simulation** Für den in der Expansion Phase ausgewählten Knoten wird eine Simulation gestartet. Es werden zufällig Actions ausgewählt und auf den Game State angewendet, bis eine vorgegebene Simulationstiefe (Rolloutdepth) oder das Spielende

erreicht wurde. Das Rollout entspricht dem vom MC Agent. (Chaslot, 2010, S. 22) (Świechowski u. a., 2021, S. 7)

**Backpropagation** Die Backpropagation Phase wird erreicht, wenn eine Simulation beendet wurde. Es wird allen Elternknoten, die zu diesem Kind geführt haben, der erreichte Reward des End-Knotens der Simulation addiert. Es werden außerdem Statistiken des Knotens verändert, z.B. Anzahl der Besuche. (Chaslot, 2010, S. 23) (Świechowski u. a., 2021, S. 7)

Die Anzahl an Wiederholungen des Prozess, die Baumtiefe und die Länge der zufälligen Simulation lassen sich über die Parameter Iterations, Treedepth und Rolloutdepth bestimmen. Aufgrund der Behandlung nichtdeterministischer Ereignisse wird für MCTSE erwartet, unter den Agenten die besten Ergebnisse in Blackjack zu erzielen.

### 3.5.4 Temporal Difference (TD)

Temporal Difference ist eine spezielle Form des Reinforcement Learnings, welche seit dem Lernerfolg 1994 im Spiel Backgammon (Tesauro (1994)) stark an Popularität im Forschungsbereich des Game Learnings gewonnen hat. Beim Temporal Difference Learning wird in Betracht gezogen, dass eine Spielsituationen, auch wenn der Endzustand des Spiels noch nicht erreicht wurde, Potenzial zum Sieg oder zur Niederlage beinhaltet. Das Abschätzen zukünftiger Rewards für eine gegebene Spielsituation und die Korrektur der vergangenen Schätzungen zu einem späteren Zeitpunkt stellen den Lernprozess und somit den Kern von Temporal Difference Learning dar.

#### 3.5.4.1 Funktionsweise

Der TD Agent verwendet eine Spielfunktion (Value Function) $V(s_t)$ um Spielsituationen zu bewerten, wobei $s_t$ ein Game State $s$ zu einem bestimmten Zeitpunkt $t$ sei. Im Idealfall würde eine erlernte Spielfunktion für jeden $s_t$ eine korrekte Abschätzung der zukünftigen Rewards angeben und somit der TD Agent optimal spielen. Beim Lernen der Spielfunktion tritt das Problem auf, dass für viele Spiele der Reward $r(s_t)$ erst zu einem terminalen Game State $s_N$ feststeht. Um dieses Problem zu lösen wird ein Fehlersignal definiert:

$$\delta_t = r(s_{t+1}) + \gamma V(s_{t+1}) - V(s_t) \tag{3.1}$$

> „The error signal vanishes if $V(s_t)$ reaches the target $r(s_{t+1}) + \gamma V(s_{t+1})$. The algorithm waits for the next state $s_{t+1}$ and looks if for that state the reward or the value function is known. If so, it changes $V(st)$ in that direction.“ (Konen, 2015, S. 4)

Das Anpassen der Spielfunktion $V(s_t)$ stellt einen Lernschritt des TD Agenten dar. Der Parameter $\gamma$ (Discount-Faktor), üblicherweise $= 0.9$, verringert den Einfluss einer zukünftigen Spielfunktion (z.B. $V(s_{t+10})$) auf die aktuelle Spielfunktion $V(s_t)$, da nicht garantiert ist, dass aus einem Game State $s_t$ der Game State $s_{t+10}$ erreicht wird.

Ein weiteres Problem ist, dass für komplexe Spiele die Zahl an möglichen Game States $s_t$ so groß wird, dass diese weder in Gänze gespeichert noch beim Lernen hinreichend besucht werden können. Um dieses Problem zu lösen, definiert man eine Funktion $f(w; s_t)$ mit freien Parametern $w$ (weigths), sodass

$$V(s_t) = f(w; s_t) \tag{3.2}$$

bestmöglich approximiert wird. (Konen (2015)) (Konen u. Bartz-Beielstein (2008))

Üblicherweise wird $f(w; s_t)$ als neuronales oder lineares Netz umgesetzt. Außerdem können Game States nach ihrenen Merkmalen $g(s_t)$ generalisiert werden. Das verwendete Netz soll somit z.B. für einen Input eines Merkmals $g(s_t)$ das Ergebnis der Valuefunction $V(s_t)$ approximieren. Der in dieser Arbeit verwendete TD-N-Tuple Agent erstellt die Merkmale $g(s_t)$ zufällig und benutzt diejenigen Merkmale mit den besten Ergebnissen. Es besteht jedoch auch die Möglichkeit Merkmale für das jeweilige Spiel zu definieren.

### 3.5.5 Basic Strategy Blackjack Agent(BSBJA)

Für Blackjack im GBG wurde der spielspezifische Basic Strategy Blackjack Agent (BSBJA) implementiert. BSBJA setzt die Basic Strategy um und bildet somit den Maßstab perfekten Spielens. Im Rahmen der Implementierung von Blackjack in GBG stellt ein Agent bei Umsetzung der Basic Strategy die höchste Spielstärke dar.

Die Basic Strategy wird von WizardsOfOdds in folgender Konfiguration bezogen: [3]

- Decks = 4 or more
- Soft 17 = dealer stands
- Double after Split = allowed
- Surrender = Allowed with any Dealer upcard
- Dealer peek = Dealer peeks for BlackJack

Die resultierende Tabelle ist in Abbildung 3.7 zu sehen. In der erste Spalte wird die Handsumme des Spielers angegeben und in der Kopfzeile die offene Karte des Dealers. Reduziert man einen Game State von Blackjack auf die Handsumme des Spielers gegen die offene Karte des Dealers, können 350 verschiedene Spielsituationen auftreten, in denen der Agent eine Entscheidung treffen muss. Für den BSBJA wurde eine Lookup-Tabelle in Form eines zweidimensionalen Arrays angelegt, welche die Basic Strategy repräsentiert. Für jeden Spielzustand, in dem bereits eine Hand ausgeteilt wurde, kann der BSBJA anhand der eigenen Handsumme und der offenen Karte des Dealers den besten Spielzug in der Tabelle mit einer konstanten Geschwindigkeit O(1) nachschlagen.

[3] https://wizardofodds.com/games/blackjack/strategy/calculator/

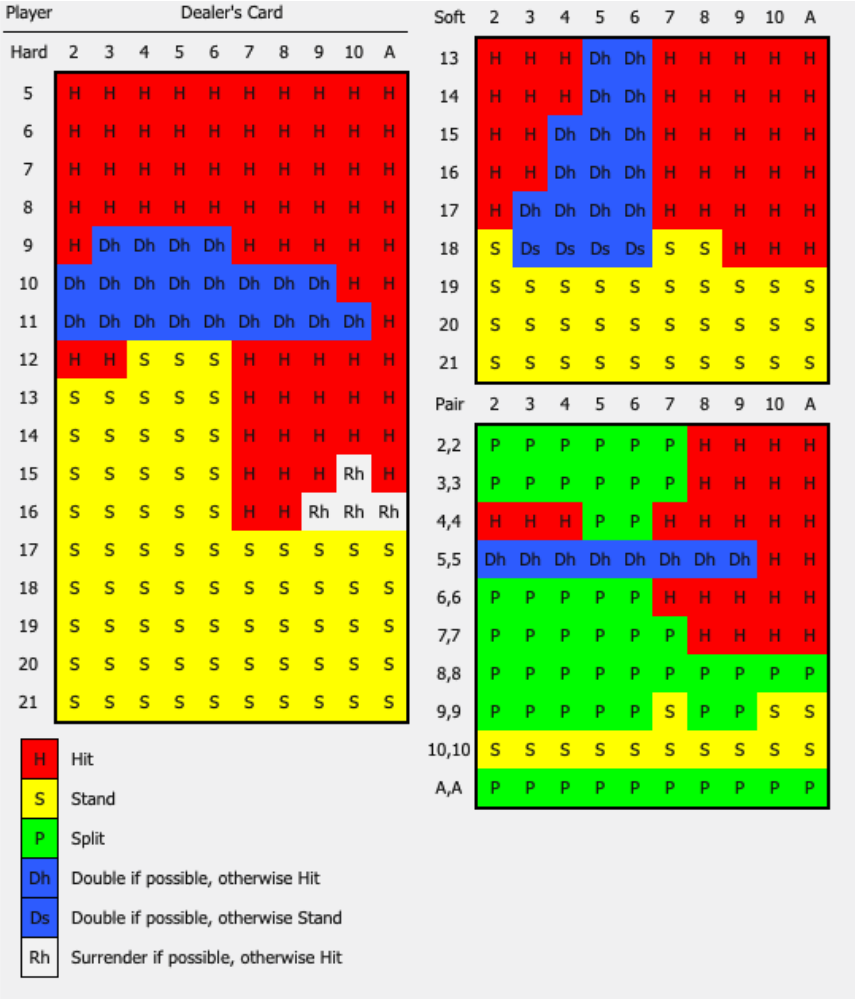


Abbildung 3.7: Basic Strategy Tabelle

### 3.5.6 Validierung der Basic Strategy

Bevor der BSBJA als Maßstab perfekten Spielens für Blackjack in GBG gilt, muss die verwendete Basic Strategy Tabelle validiert werden. Hierfür wurde eine JUnit Testumgebung erstellt. 20 Spielsituationen aus der Basic Strategy wurden ausgewählt und die Korrektheit des vermeintlich besten Spielzugs getestet. Bei der Auswahl der Spielsituation wurden möglichst heterogene Spielsituationen ausgewählt (die Hand 2,3 wird genauso wie die Hand 2,4 gespielt).

#### 3.5.6.1 Test zur Validierung

Um ein Spielzug für eine Spielsituation bzw. Spielzustand zu bewerten, wird für einen Spielzustand s der Spielzug a gemacht. Anschließend kann der daraus resultierende Spielzustand s' nach gemachten Spielzug a betrachtet werden. Falls nach dem gezogenen Spielzug a kein Rundenende erreicht wurde, werden Spielzüge aus der Basic Strategy angewendet, bis ein Rundenende erreicht wird. Ist ein Rundenende erreicht, kann

der Payoff (Reward) der Runde abgelesen werden. Wiederholt man die beschriebene Sequenz für einen Zug (z.B. 5000 Mal), kann für diesen Spielzug ein durchschnittlicher Payoff ermittelt werden. Gibt die Basic Strategy einen Spielzug als den besten Spielzug an, so muss sein durchschnittlicher Payoff höher als der durchschnittliche Payoff aller anderen verfügbaren Züge sein. Folglich muss für alle Spielzüge in einem Game State der durchschnittliche Payoff gemessen und mit dem durchschnittlichen Payoff aller anderen Spielzüge verglichen werden.

#### 3.5.6.2 Testergebnis

Für die 20 getesteten Spielsituation ergaben die Messungen, dass der von der verwendeten Basic Strategy angegebene Spielzug für die entsprechende Spielsituation den höchsten durchschnittlichen Payoff aufweist. Somit gilt die Basic Strategy vorerst als validiert. Im Rahmen weiterer Arbeiten sollte jeder angegebene Spielzug der Basic Strategy auf Korrektheit überprüft werden. Das positive Ergebnis zeichnet sich für die meisten Spielsituationen bei $10^4$ Spielrunden ab. Für einige Spielsituationen (z.B. player{3, 3} gegen dealer{8, X}) werden mehr Spielrunden zum Bilden des durchschnittlichen Payoffs benötigt, um ein konsistentes und positives Ergebnis zu erhalten. In folgender Tabelle werden die Testergebnisse für die Spielsituation player{3, 3} gegen dealer {8, X} mit verschiedenen Iterationen dargestellt. Die verwendete Basic Strategy gibt für diese Spielsituation den Spielzug Hit als besten Spielzug an.

| gespielte Runden | Stand | Hit | Doubledown | Split | Surrender |
|---|---|---|---|---|---|
| $10^3$ | −5.18 | −2.48 | −10.28 | −2.08 | −5.0 |
| $10^4$ | −4.982 | −2.211 | −10.14 | −2.523 | −5.0 |
| $10^5$ | −5.1162 | −2.1883 | −10.0112 | −2.4653 | −5.0 |
| $10^6$ | −5.11002 | −2.174 | −10.06582 | −2.28793 | −5.0 |
| $10^7$ | −5.10287 | −2.171904 | −10.019006 | −2.306416 | −5.0 |

Tabelle 3.1: Durchschnittlicher Payoff der möglichen Spielzüge, abhängig von der Anzahl an gespielten Runden, für die Spielsituation player{3, 3} gegen dealer{8, X}

In der Tabelle lässt sich erkennen, dass mit $10^3$ Spielrunden der Test scheitert. Hier hat der Spielzug Split den höchsten durchschnittlichen Payoff. Ab $10^4$ Spielrunden liefert der Test das erwartete Ergebnis. An den Ergebnissen für den Spielzug Split lässt sich außerdem erkennen, dass Messungen in Blackjack für manche Fälle sehr viele Wiederholungen benötigen, um sich einem konstanten Ergebnis anzunähern.

# 4 Evaluation

## 4.1 Evaluationstrategien

Viele implementierte Spiele im GBG setzen ihre Evaluation für Agenten um, indem sie verschiedene Agenten gegeneinander antreten lassen. Somit kann das Stärkeverhältnis zwischen Agenten bestimmt werden. Ist zusätzlich ein optimal spielender spielspezifischer Agent implementiert, können die allgemeinen Agenten an diesem "Benchmark Agenten" gemessen werden. In Blackjack ist dies nicht ohne Weiteres möglich, da Agenten unabhängig voneinander gegen den Dealer spielen und nicht gegeneinander spielen können. Es müssen somit Ausprägungen gefunden werden, die für den zu evaluierenden Agent im 1-Spieler Modus gemessen werden können.

### 4.1.1 Durchschnittlicher Payoff

Eines der aussagekräftigsten Merkmale zur Spielstärke stellt der durchschnittliche Payoff eines Agenten dar. Dieser kann auch am effektivsten die finale Gesamtstärke eines Agenten abbilden. Jedoch kann es wie bei der Validierung der Basic Strategy für den Spielzug Split sehr viele Runden benötigen, bis der durchschnittliche Payoff zu einem stabilen Wert konvergiert. Wie in Abbildung 4.1 zu erkennen, erreicht der BSBJA erst zwischen $10^5$ und $10^6$ gespielten Runden den erwarteten, geringen, negativen Payoff.

Ein besonderes Problem stellt der Agent MCTSE mit dieser Evaluationsmethode dar. Diese benötigt abhängig von den für ihn gesetzten Parameter mehrere Sekunden Rechenzeit, um einen Spielzug zu wählen. Für eine einzige Messung, in der 1000 Runden gespielt werden, benötigt der Agent über eine halbe Stunde. Jede Messung in dieser Arbeit wird mindestens zehnmal ausgeführt und aus diesen Messungen werden der Durchschnitt und die Standardabweichung ermittelt. Somit würde eine gesamte Messung mit einer bestimmten Parameterkombination für MCTSE mehrere Stunden dauern. Aus diesem Grund wurde die Anzahl der Runden zur Ermittlung des durchschnittlichen Payoff auf 50 reduziert. Diese geringe Anzahl an Runden liefert nach 10 Messungen trotz Bildung des Durchschnitts und der Standardabweichung ein wenig aussagekräftiges Ergebnis. Darüber hinaus stellt der durchschnittliche Payoff eine Gesamtstärke dar, bei der die möglichen Stärken oder Schwächen unerkannt bleiben können. Ein Beispiel hierfür ist ein Agent, der immer den besten Spielzug Split wählt, jedoch selten den besten Spielzug Stand wählt. Die Stärke des Agenten, immer den besten Spielzug Split zu wählen, könnte von seiner Schwäche, selten den Spielzug Stand zu wählen, überdeckt werden.

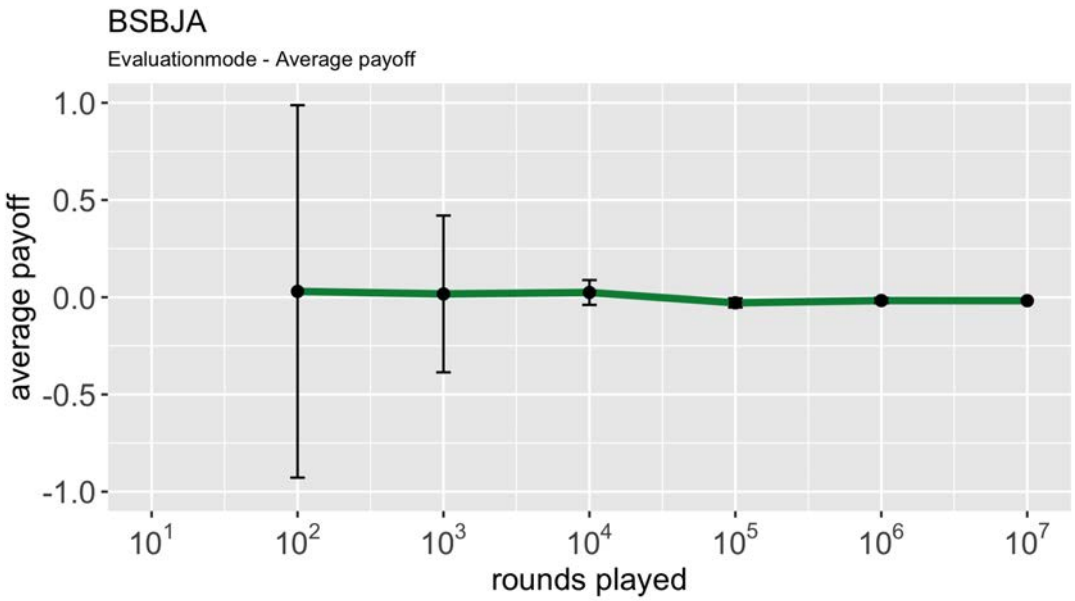


Abbildung 4.1: Durchschnittlicher Payoff BSBJA (mit Fehlehrbalken, der Standardabweichung aus zehn Messungen)

### 4.1.2 Situationen aus der Basic Strategy

Diese Evaluationsstrategie wurde für eine schnellere Evaluation entwickelt. Hierfür wurden 23 vorgefertigte Spielsituationen erstellt, in denen der Agent einen Spielzug wählt. Anschließend wird geprüft, ob der gewählte Spielzug des Agenten dem Spielzug gleicht, welchen die Basic Strategy als besten Spielzug angibt. Das Verhältnis aus den Entscheidungen, die der Basic Strategy gleichen und solchen, die es nicht tun, stellt das Evaluationsergebnis dar (0 bis 1). Bei der Auswahl der Spielsituation wurden möglichst heterogene Spielsituationen ausgewählt. Von dieser Evaluation wird erwartet, dass sich beim Erzielen eines hohen Werts in der Evaluation des Agenten sein durchschnittlicher Payoff dem des BSBJA annähert.

### 4.1.3 Zufällige Situationen aus der Basic Strategy

Um ein Bias in der Evaluationsstrategie “Situationen aus der Basic Strategy” aus Abschnitt 4.1.2 auszuschließen, werden in dieser Evaluationsstrategie zufällige Spielsituationen generiert. Für die zufälligen Spielsituationen wird ebenfalls geprüft, welcher Anteil der getätigten Spielzüge mit denen der Basic Strategy übereinstimmt.

### 4.1.4 Einfache Spielsituationen

Die folgenden beiden Evaluationsstrategien repräsentieren ein Grundverständnis von Blackjack und konzentrieren sich auf die Spielzüge Hit und Stand. Hat ein Agent einen sehr geringen Handwert (z.B. player$\{2, 3\} = 5$), sollte der offensichtliche Spielzug Hit sein. Keine Karte, welche der Agent durch den Spielzug Hit erhalten kann, verschlechtert die Gewinnchancen gegenüber dem Spielzug Stand. Hat ein Agent eine hohen

Handwert (z.B. player{K, 10} = 20), sollte der offensichtliche Spielzug Stand sein. Lediglich ein Ass könnte die Gewinnchancen dieser Hand noch verbessern. Alle anderen Karten reduzieren die Gewinnchancen dieser Hand auf Null. Die gewonnen Evaluationsergebnisse stellen somit einen sehr spezifischen Fähigkeitenbereich eines Agenten wieder. Somit kann dieser spezifisch gewählte Fähigkeitenbereich des Agenten als Stärke oder Schwäche isoliert werden.

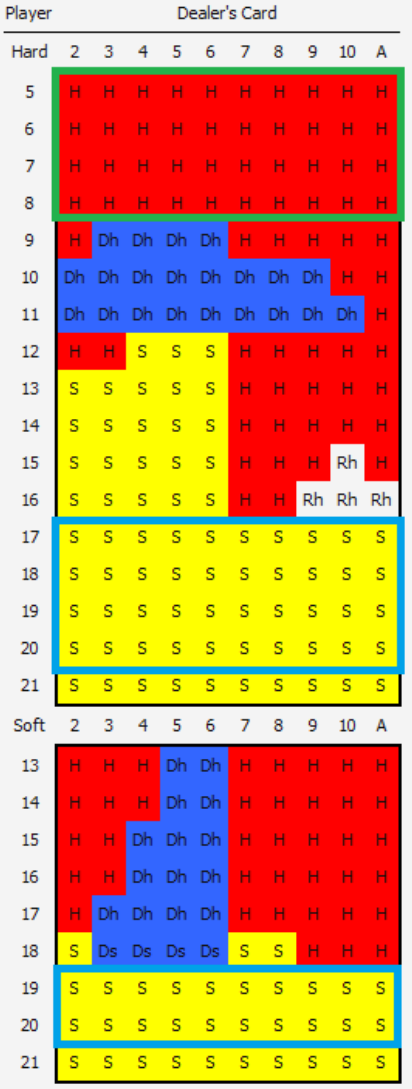


Abbildung 4.2: Basic Strategy Tabelle markiert

### 4.1.5 Einfache Spielsituationen mit bestem Spielzug Hit

Der zu evaluierende Agent spielt 40 Spielsituationen, in denen Hit der beste Spielzug ist und der zu erwartende Payoff des Spielzugs Hit deutlich höher gegenüber allen anderen Zügen ist. Das Ergebnis dieser Evaluation ist das prozentuale Verhältnis aus den Situationen, in welchen der Agent den Spielzug Hit gewählt hat und den Situationen, in denen er ihn nicht gewählt hat (0-1). Für die 40 Spielsituationen wurde der korrespondierende Teil in der Basic Strategy in Abbildung 4.2 grün markiert.

### 4.1.6 Einfache Spielsituationen mit bestem Spielzug Stand

Der zu evaluierende Agent spielt 60 Spielsituationen, in denen Stand der beste Spielzug ist und der zu erwartende Payoff des Spielzugs Stand deutlich höher gegenüber allen anderen Zügen ist. Das Ergebnis dieser Evaluation ist das prozentuale Verhältnis aus den Situationen, in welchen der Agent den Spielzug Stand gewählt hat und den Situationen, in denen er ihn nicht gewählt hat (0-1). Für die 60 Spielsituationen wurde der korrespondierende Teil in der Basic Strategy in Abbildung 4.2 blau markiert.

## 4.2 Der Parameter StopOnRoundOver

Wie im Abschnitt Rundenende 3.4.2 angesprochen, ist Blackjack ein episodisches Spiel mit voneinander unabhängigen Runden. Für (vorerst Monte Carlo) Agenten wurde darüber nachgedacht, ob ein Agent eine Simulation, konkret das zufällige Rollout, bis zu einem erreichten Rundenende oder über das Rundenende hinaus machen sollte. Es wurde sich auf die Annahme geeinigt, dass ein Rollout über das Rundenende hinaus für Blackjack nicht von Vorteil ist. Aufgrund dieser ersten Annahme wurden die allgemeinen Agenten im GBG dahingehend generalisiert, sodass diese ihre Rollouts nur bis zum Erreichen eines Rundenendes ausführen. Dieses Verhalten ist jedoch optional und kann über den Parameter StopOnRoundOver eingestellt werden. Zur Prüfung dieser Annahme wird für Blackjack mit dem MC-N Agenten im Folgenden ein Experiment durchgeführt.

### 4.2.1 MC Experiment StopOnRoundOver true versus false

#### 4.2.1.1 Aufbau

Es werden Messungen bzw. Evaluationen für MC-N sowohl für den Parameter StopOnRoundOver mit dem Wert true und mit dem Wert false ausgeführt und gegenübergestellt. Eine gesamte Messung für eine Parameterkombination besteht aus 10 Messungen, aus denen jeweils das arithmetische Mittel sowie die Standardabweichung ermittelt werden. Für das Experiment wird außerdem die Rolloutdepth sukzessive erhöht. Der Parameter Iterations bleibt konstant auf 1000. Außerdem werden alle vier Evaluationsmethoden aus dem Abschnitt 4.1 verwendet.

#### 4.2.1.2 Prognose

Es soll die Annahme geprüft werden, ob eine Simulation, konkret das zufällige Rollout, über das Rundenende hinaus keinen Vorteil bringt. Konkret wird erwartet, dass bei einer Simulation über das Rundenende hinaus entweder die Leistung des Agenten gleich bleibt oder sogar schlechter wird. Bei gleicher Spielstärke würde eine Simulation über das Rundenende hinaus Rechenleistung verschwenden. Folglich wäre das Stoppen am Rundenende durch die gewonnene Rechenleistung von Vorteil. Bei Verschlechterung der Spielstärke wären sogar Rechenleistung und Spielstärke beim Stoppen am Rundenende gewonnen. Eine Verschlechterung der Spielstärke ließe sich damit erklären, dass die Strategie einer Runde in Blackjack unabhängig von vergangenen und zukünftigen Runden gewählt werden sollte. Eine Runde stellt unter der Voraussetzung, dass keine fortlaufenden Informationen wie beim Kartenzählen gesammelt werden, ein in sich geschlossenes System dar. Die Simulation einer folgenden Runde würde somit ein Störsignal darstellen. Ein mögliches Szenario wäre eine Runde, die in der Simulation gewonnen wird, worauf eine Runde folgt, welche verloren wird. Der positive Reward, der in der gewonnen Runde erreicht wird, wird durch die darauffolgende verlorene Runde neutralisiert.

#### 4.2.1.3 Visuelle Repräsentation der Messungen

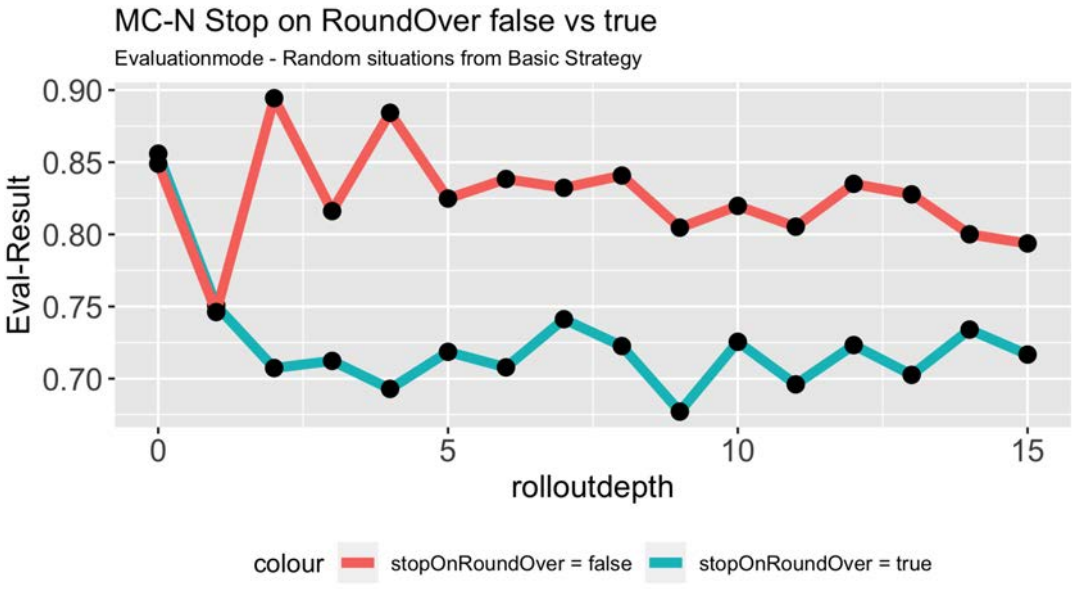


Abbildung 4.3: StopOnRoundOver Messung 1

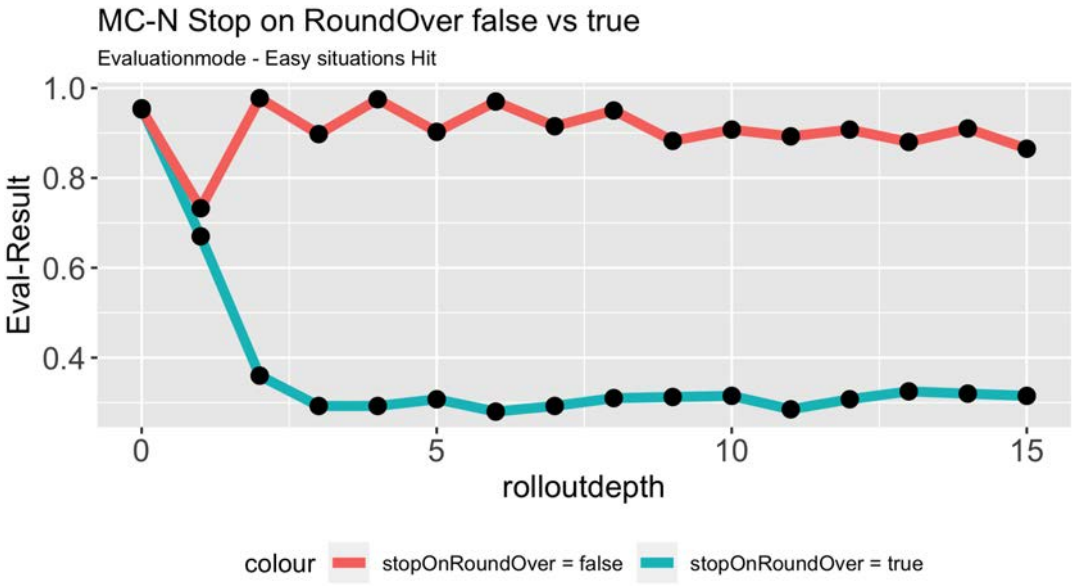


Abbildung 4.4: StopOnRoundOver Messung 2

#### 4.2.1.4 Interpretation der Messdaten

Entgegen der Prognose ist in Abbildung 4.3 zu sehen, dass die Evaluationsergebnisse mit einem Rollout über das Rundenende hinaus besser ausgefallen sind. Das Evaluationsergebnis aus Abbildung 4.4 wirkt unterstützend zum Verständnis von Abbildung 4.3. Hier lässt sich deutlich erkennen, dass bei einer Simulation mit StopOnRoundOver = true der Agent Probleme hat, Hit als besten Spielzug zu identifizieren. Diese Schwäche wirkt sich auf seine allgemeine Spielstärke aus. Weitere Messungen mit sukzessiver Erhöhung des Parameters Iterations haben ergeben, dass die erkannte Schwäche weiterhin besteht. Für den Agenten MCTSE konnte dasselbe Problem erkannt werden, wobei hier weniger Messungen aufgrund der hohen Rechenzeiten von MCTSE durchgeführt wurden. Kurzfristig konnte für diese unerwarteten Ergebnisse keine schlüssige Begründung gefunden werden. Es wurde vermutet, dass die fehlerhafte Darstellung des Rewards dieses unerwartete Verhalten verursacht. Nach Behebung der fehlerhaften Darstellung des Rewards wurde das Experiment wiederholt, jedoch ohne erkennbare Veränderung in den Ergebnissen. Eine weitere überraschende Unschlüssigkeit ließ sich in Abbildung 4.3 erkennen. Für eine Rolloutdepth von 3 sollte der Unterschied zwischen den beiden Varianten nicht so hoch ausfallen. Unter der Annahme, dass im Mittel eine Runde in Blackjack nach zwei oder drei Spielzügen beendet ist, dürften sich hier die Evaluationsergebnisse nicht so drastisch unterscheiden, da beide Varianten ähnliche Simulationstiefen haben. Ein mögliches Fehlverhalten der Blackjack Umgebung im GBG wurde ebenfalls in Betracht gezogen.

## 4.3 Evaluation der Agenten

Aufgrund der im Abschnitt 4.2.1 gewonnen Erkenntnisse werden alle Evaluationen für Monte Carlo Agenten mit dem Parameter StopOnRoundOver = false ausgeführt. Für MC-N werden die Messdaten ohne Standardabweichung dargestellt, um die Diagramme mit mehr als einer Linie übersichtlich zu halten. Es wurden keine Auffälligkeiten in der Standardabweichung erkannt.

### 4.3.1 Evaluation von MC

Im Folgenden soll die Grund- und Gesamtstärke von MC gemessen werden.

#### 4.3.1.1 Grundstärke

Aufgrund der schwer zu erfassenden Gesamtstärke eines Agenten, wird für jeden Agenten die Grundstärke durch die Fähigkeit repräsentiert, ob dieser in den einfachsten Situationen den Spielzug Hit bzw. Stand korrekt wählt. Diese Grundstärke wird mit den beiden Evaluationsstrategien “Einfache Spielsituationen mit bestem Spielzug Hit” und “Einfache Spielsituationen mit bestem Spielzug Stand” gemessen. Aufgrund der Geschwindigkeit von MC-N konnte der Parameter Iterations bis 20000 erhöht werden, um weitere Tendenzen im Spielverhalten zu erkennen. Des Weiteren konnte die Erhöhung des Parameters Iterations mit Erhöhung des Parameters Rolloutdepth kombiniert werden. Um eine faire Gegenüberstellung zu MCTSE zu erhalten, sollten jedoch

nur Werte bis iterations = 6000 zur Gegenüberstellung der Spielstärke betrachtet werden. Wie in Abbildung 4.5 und 4.6 zu sehen ist, findet MC-N in über 90% aller Fälle die offensichtlichen Spielzüge Hit und Stand. Für manche Parameterkombinationen können konstante Werte von über 95% erreicht werden. MC-N schafft es mit geringer Rechenzeit und verschiedensten Parameterkombinationen die Grundlagen von Blackjack umzusetzen. Für den Spielzug Stand scheint eine starke Erhöhung der Iterationen einen Aufwärtstrend zu erzielen.

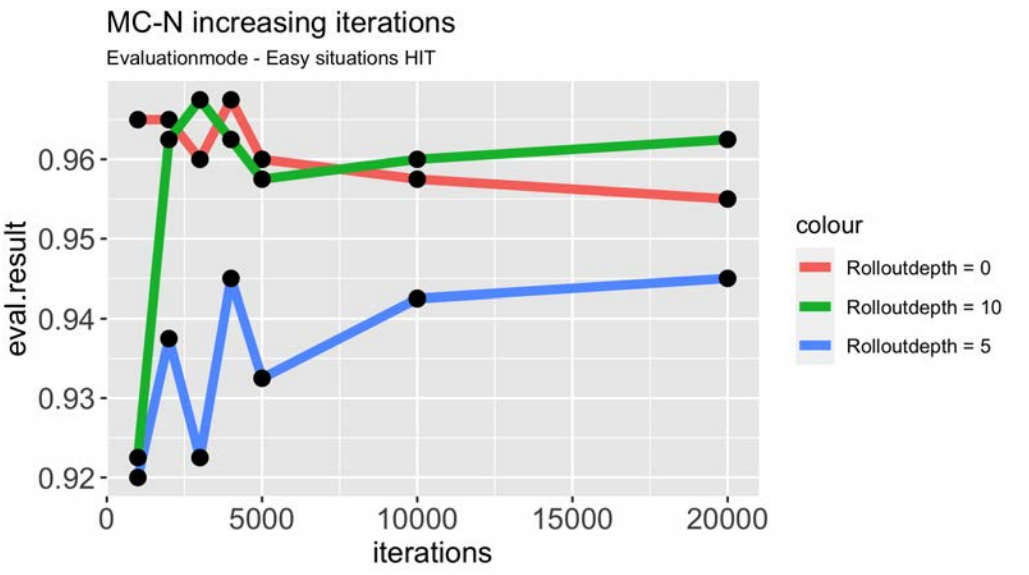


Abbildung 4.5: MC Einfache Spielsituationen mit bestem Spielzug Hit

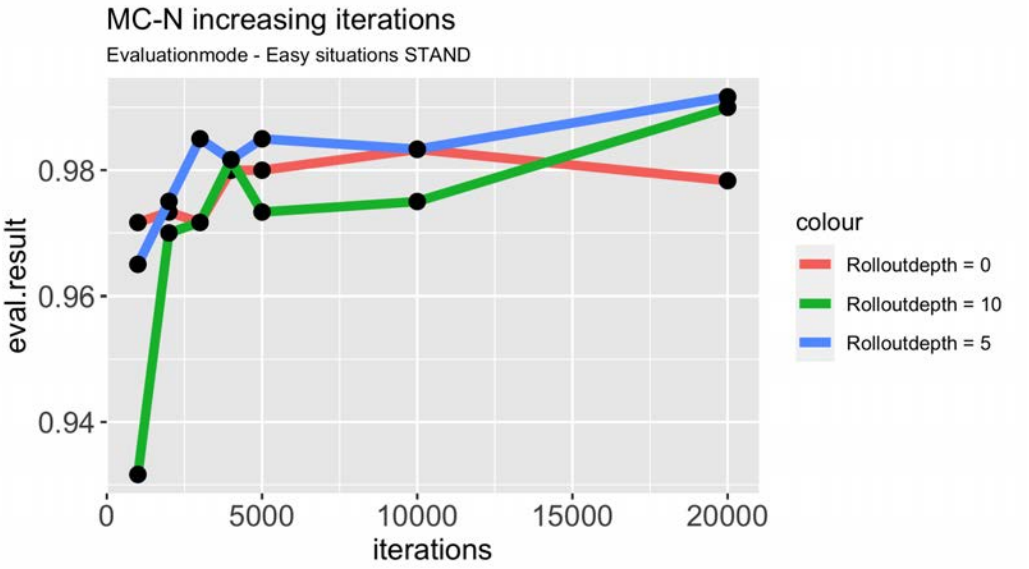


Abbildung 4.6: MC Einfache Spielsituationen mit bestem Spielzug Stand

#### 4.3.1.2 Gesamtstärke

Die Gesamtstärke eines Agenten wird über die Evaluationsstrategien "Durchschnittlicher Payoff" und "Zufällige Situationen aus der Basic Strategy" gemessen. Aufgrund der wenigen Spielrunden (50 Runden pro Messung * 10 Messungen pro Parameterkombination), aus denen das arithmetische Mittel dieser Evaluationsstrategien berechnet wurde, sollten keine finalen Schlüsse zur Gesamtstärke eines Agenten gezogen werden. Auf Grafik 4.7 ist zu sehen, dass MC mit jeder Parameterkombination annähernd konstant in über 80% der Spielsituationen den besten Spielzug findet. Das beste Ergebnis konnte, wie in Abbildung 4.7 abzulesen, mit einer Rolloutdepth von 10 erreicht werden, dies spiegelt sich jedoch nicht in Abbildung 4.8 wider. In Abbildung 4.8, dem durschnittliche Payoff, konnten für MC Werte im Bereich von $-0.5$ bis 0.6, sowie eine Standardabweichungen im Bereich von $\pm 1$ bis $\pm 2$ beobachtet werden.

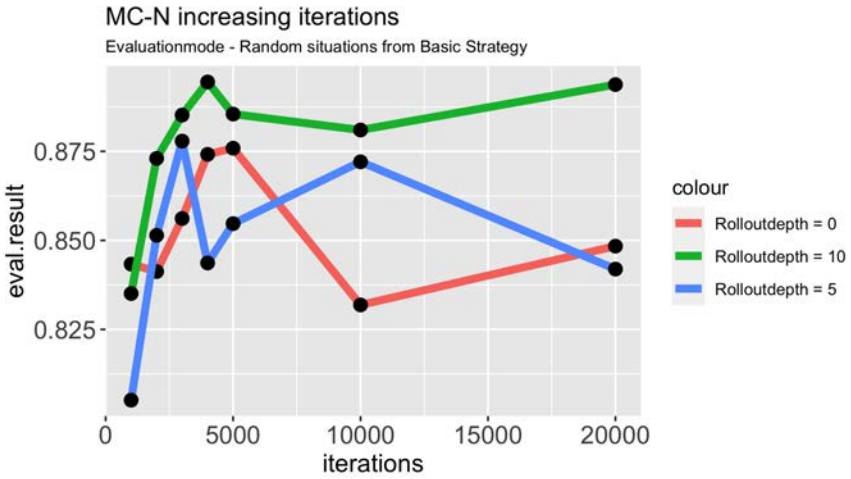


Abbildung 4.7: MC Zufällige Situationen aus der Basic Strategy

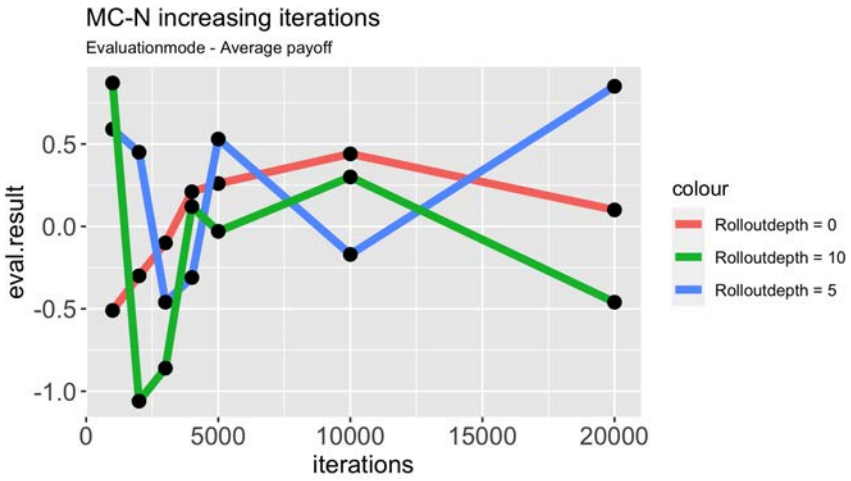


Abbildung 4.8: MC Durchschnittlicher Payoff

### 4.3.2 Evaluation von MCTSE

Im Folgenden soll die Grund- und Gesamtstärke von MCTSE gemessen werden.

#### 4.3.2.1 Grundstärke

Aus Abbildung 4.9 lässt sich ableiten, dass MCTSE stark von steigenden Iterationen profitiert. Hier können ab 5000 Iterationen Werte von über 96% erreicht werden. In Abbildung 4.10 hingegen sinken die Evaluationsergebnisse der einfachen Spielsituationen Hit und Stand bei steigender Rolloutdepth. Aus Abbildung 4.11 lässt sich erkennen, dass mindestens eine Treedepth von 2 erwünscht ist, um gute Resultate für den Spielzug Hit zu erkennen. Die besten Resultate können mit einer Treedepth zwischen 5 und 9 erreicht werden. MCTSE schafft es wie MC die Grundlagen von Blackjack umzusetzen, jedoch konnte bei den Evaluationsmessungen festgestellt werden, dass MCTSE deutlich höhere Rechenzeiten benötigt.

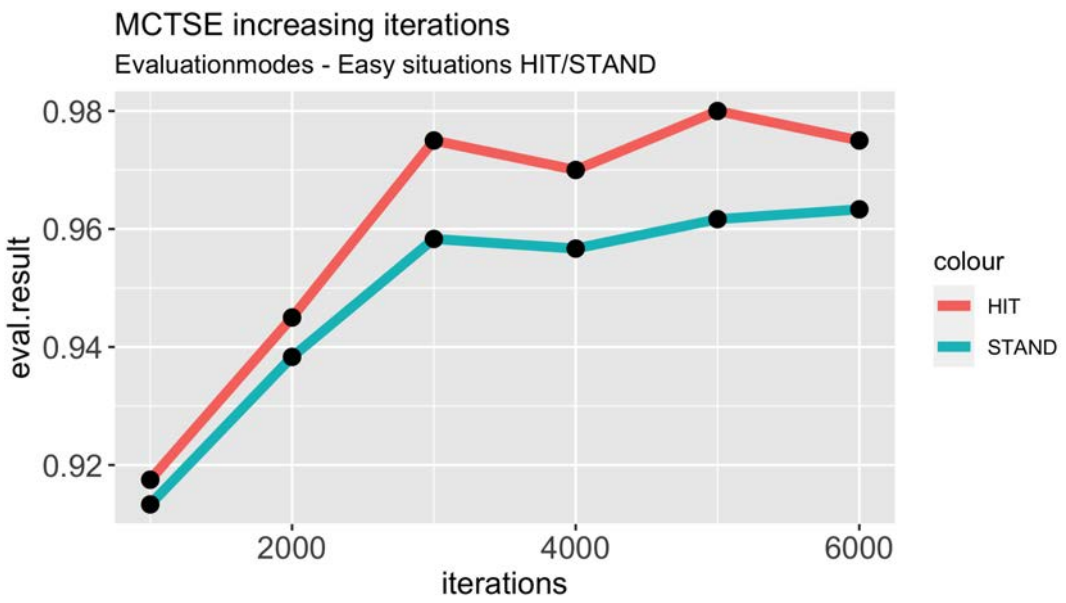


Abbildung 4.9: MCTSE Einfache Spielsituationen mit bestem Spielzug Hit/Stand mit steigenden Iterationen

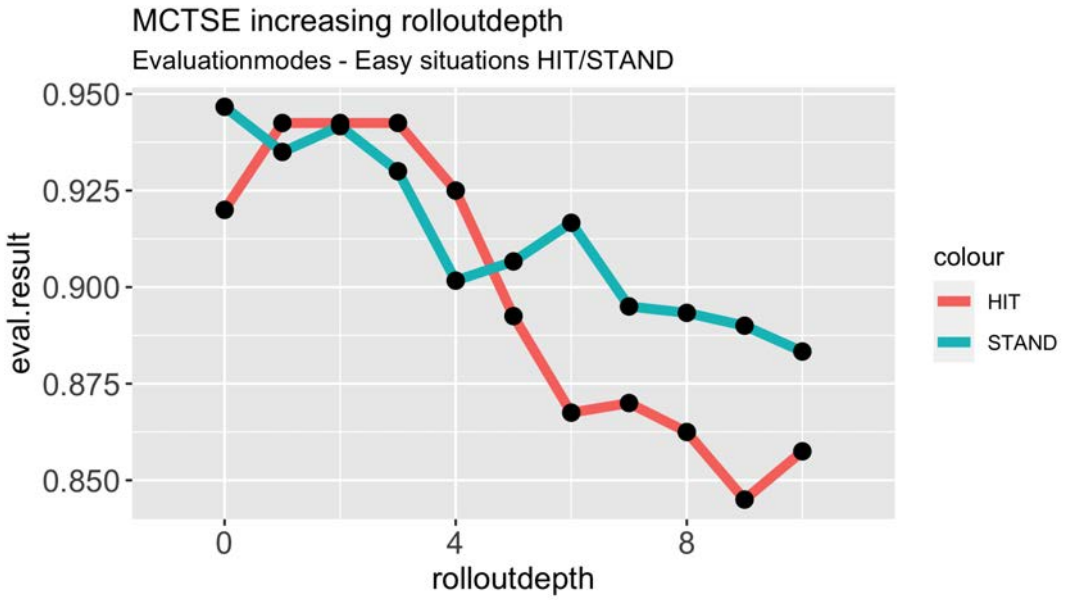


Abbildung 4.10: MCTSE Einfache Spielsituationen mit bestem Spielzug Hit/Stand mit steigender Rolloutdepth

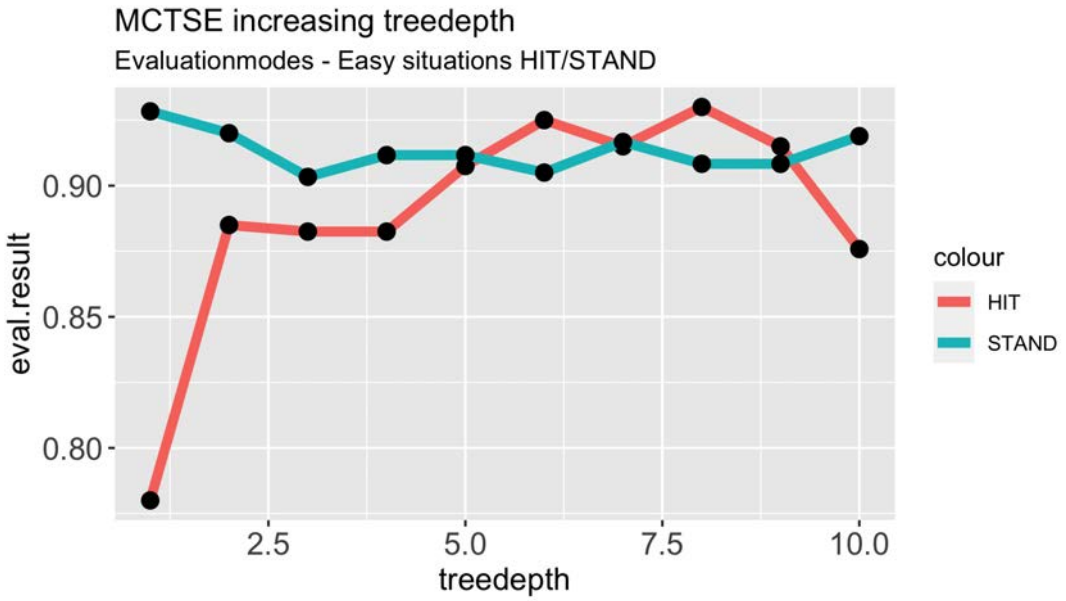


Abbildung 4.11: MCTSE Einfache Spielsituationen mit bestem Spielzug Hit/Stand mit steigender Treedepth

#### 4.3.2.2 Gesamtstärke

In Abbildung 4.13 ist zu beobachten, dass MCTSE für zufällige Situationen aus der Basic Strategy Werte über 80% erzielen konnte. Der Höchstwert liegt bei über 87%. Wie auch für die Evaluation der Grundlagen ist hier zu erkennen, dass das Erhöhen der Iterationen zu besseren Ergebnissen führt. In Abbildung 4.12 erzielt MCTSE mit steigenden Iterationen einen besseren durchschnittlichen Payoff. Bei der Messung des durchschnittlichen Payoff konnten Ergebnisse im Bereich zwischen $-0.6$ und 0.1, sowie eine Standardabweichungen im Bereich im Bereich von $\pm 1.2$ bis $\pm 2$ erhoben werden.

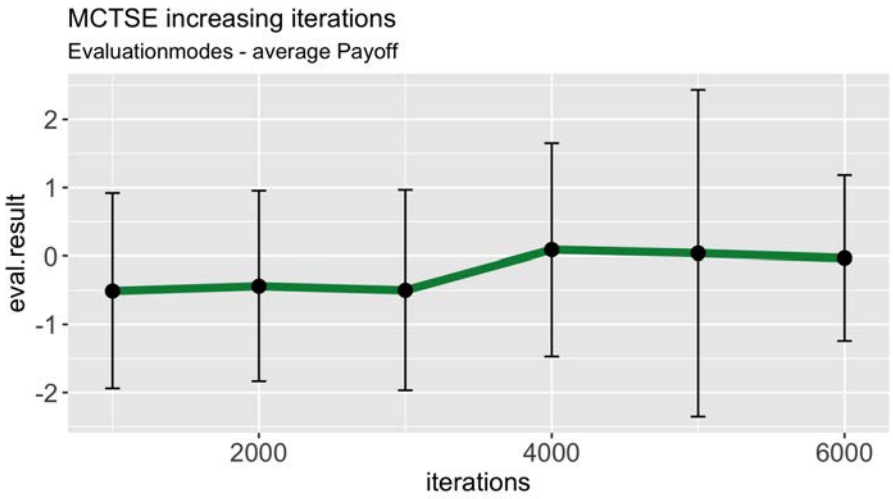


Abbildung 4.12: MCTSE durchschnittlicher Payoff mit steigenden Iterationen (mit Fehlehrbalken, der Standardabweichung aus zehn Messungen)

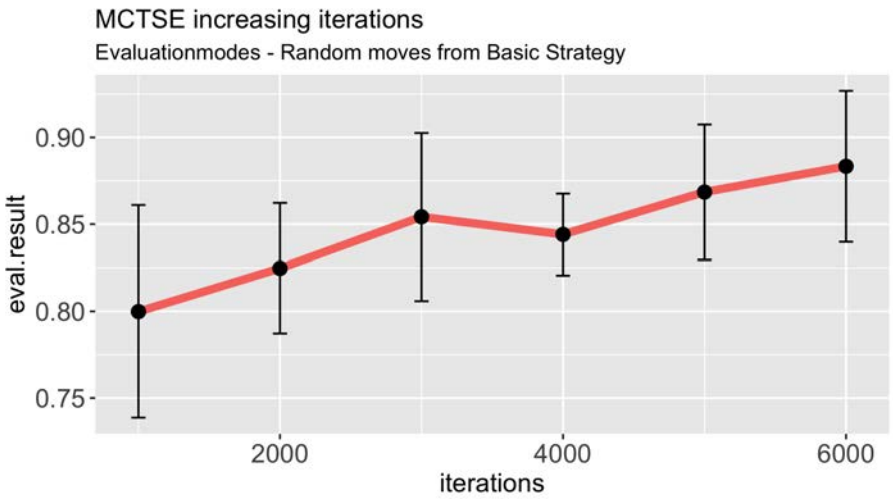


Abbildung 4.13: MCTSE Zufällige Situationen aus der Basic Strategy mit steigenden Iterationen (mit Fehlehrbalken, der Standardabweichung aus zehn Messungen)

### 4.3.3 Evaluation von TD-N-Tuple 4

Für den TD-N-Tuple Agent konnte mit verschiedensten Parameterkombination keinerlei Lernerfolg für Blackjack erkannt werden. In Abbildung 4.14 hat der Agent 500000 Runden trainiert und wurde mit den Evaluationsstrategien "Einfache Situationen mit besten Spielzug Hit" und "Einfache Situationen mit besten Spielzug Stand" evaluiert. Der Agent hat es nicht geschafft die Grundlagen von Blackjack zu erlernen. Aufgrund der fehlenden Grundlagen wurde von weiteren Evaluationen abgesehen.


Abbildung 4.14: Trainingsfortschritt TD-N-Tuple 4

# 5 Problemanalyse und Reflektion

## 5.1 Gegenüberstellung von MC und MCTSE

Im Abschnitt 4.3 wurden Evaluationsergebnisse für MC und MCTSE gesammelt und dargestellt. Aus den präsentierten Daten lässt sich ablesen, dass sowohl MC als auch MCTSE Erfolge in der definierten Grundstärke vorweisen konnten. Für die Evaluationsmethode „Einfache Spielsituationen mit besten Spielzug Hit“ konnte MCTSE einen Höchstwert mit knapp über 98% erzielen. MC lag jedoch mit einem Ergebnis von knapp über 96.5% nicht weit dahinter. Für die anderen Evaluationsmethoden, auch dem durchschnittlichen Payoff, konnten die höchsten Werte für MC gemessen werden. Folglich kann für MC gegenüber MCTSE in der erstellten Blackjack Umgebung eine höhere Spielstärke erkannt werden.

Des Weiteren stellt die lange Rechenzeit von MCTSE ein Problem dar. Für MCTSE ist durch sukzessive Erhöhung des Parameters Iterations ein Aufwärtstrend in den Evaluationsergebnissen 4.13 und 4.12 zu erkennen. Dieser Aufwärtstrend lässt sich jedoch im Rahmen dieser Arbeit nicht weiter untersuchen, da ein Satz von zehn Messungen schon für 6000 Iterationen mindestens sechs Stunden dauert. Für MCTSE ist somit das Testen von verschiedenen Parameterkombinationen zur Verbesserung von Evaluationsergebnissen erschwert. Eine im Verlaufe dieser Arbeit erdachte Theorie ist, dass für Blackjack der Spielbaum von MCTSE in der Höhe reduziert und in der Breite ausgedehnt werden sollte. Dadurch könnten für Expectimax-Knoten mehr Chance-Knoten angehängt werden und somit besser über die möglichen Ausgänge eines nichtdeterministischen Ereignisses gemittelt werden. Jedoch kann auch die Manipulation der gegeben Baumstruktur von MCTSE aufgrund langer Rechenzeiten nicht effizient untersucht werden. Schlussendlich kann das hoch vermutete Potenzial von MCTSE in dieser Arbeit nicht ausschöpfend bestätigt werden. Für MC konnten neben den besseren Evaluationsergebnissen auch akzeptable Rechenzeiten vermerkt werden.

## 5.2 Der Parameter StopOnRoundOver

Nach einer Fehlersuche in der Blackjack Umgebung konnte ein Bug als Grund für die Ergebnisse des StopOnRoundOver Experiments mit hoher Wahrscheinlichkeit ausgeschlossen werden. Im Verlauf der Arbeit konnte jedoch eine logische Erklärung für die Ergebnisse entgegen der gemachten Prognose gefunden werden. Hierfür betrachten wir den einfachen Game State $s_0$ = [playerHand{2,3}=5, dealerHand{6, X} = 6] und machen folgende Beobachtungen:
**1)** Wird in s0 der Spielzug Stand ausgeführt, folgt kein zufälliges Rollout, da nach Ausführung des Spielzugs Stand ein Rundenende erreicht wird. [1]
**2)** Wird in s0 der Spielzug Hit ausgeführt, folgt ein zufälliges Rollout bis zum Rundenende.
**3)** Ein Rollout von einem Monte Carlo Agenten besteht aus zufällig gewählten Spielzügen. Konkret wird im GBG das Rollout sogar von einem Random Agenten ausgeführt.
**4)** Ein zufällig spielender Agent erzielt im Mittel einen negativen durchschnittlichen Payoff (vgl. Abbildung 5.1).
Die Beobachtungen **1)** und **2)** zeigen, dass das Stoppen eines Rollouts am Rundenende den Nebeneffekt verursacht, dass auf verschiedene Spielzüge unterschiedlich lange Rollouts folgen.
Die Beobachtungen **3)** und **4)** zeigen, dass das zufällige Rollout eines Monte Carlo Agenten in Blackjack einen im Mittel negativen durchschnittlichen Payoff erzielt.
Aus diesen Beobachtungen lässt sich darauf schließen, dass ein Spielzug (bspw. Hit), auf welchen ein zufälliges Rollout folgt, im Mittel schlechter bewertet wird als ein Spielzug (bspw. Stand), auf welchen kein zufälliges Rollout folgt. Um dieses Phänomen ins Extreme zu führen, ist auf Abbildung 5.2 eine Gegenüberstellung des Spielzugs Hit (mit steigender Rolloutdepth und einem Rollout über ein Rundenende hinaus) und dem Spielzug Stand (ohne folgendem Rollout) bewertet durch MC dargestellt. Folglich entsteht für den Parameter StopOnRoundOver = true für verschiedene Spielzüge ein unfaires Bewertungsverhältnis, da auf die Spielzüge Stand, Surrender und Double Down kein Rollout folgt, auf die Spielzüge Hit und Split jedoch schon.

Dieses Problem könnte abgeschwächt werden, indem das Rollout nicht mehr aus zufälligen Spielzügen bestünde, die sich zu einem starken negativen Payoff mitteln, sondern eine Heuristik umsetzt, welche im Mittel ein Payoff nahe Null erzielt. Dies lässt sich in Tabelle 5.1 erkennen, in welcher der durchschnittliche Payoff aus $10^7$ Runden für die Spielzüge Hit und Stand in $s_0$ gebildet wird. Dabei wurde für den Spielzug Hit sowohl ein Rollout mit zufälligen Spielzügen als auch ein Rollout mit optimalen Spielzügen ausgeführt.

Außerdem ist aus der Tabelle zu entnehmen, dass sich der durchschnittlichen Payoff des Spielzugs Hit mit einem zufälligen Rollout um $\approx 2,46$ gegenüber dem Rollout mit optimaler Strategie verschlechtert. Der beste erwartete Reward des Spielzugs Hit für den Game State $s_0$ ergibt sich aus dem Rollout mit optimaler Strategie. Hingegen ergibt sich der beste erwartete Reward für den Spielzug Stand lediglich aus der

[1] Hier gilt es zwischen Simulation und Rollout zu unterscheiden. Nach dem Spielzug Stand werden die Karten des Dealers vervollständigt (Simulation). Ein Rollout im Kontext von Monte Carlo Agenten meint jedoch, (zufällige) Spielzüge auszuwählen und anzuwenden, um den Game State in den folgenden zu überführen. Ein Rollout ist somit Teil der Simulation.

| Stand | Hit (zufälliges Rollout) | Hit (optimales Rollout) |
|---|---|---|
| −1.534224 | −2.458864 | −0.011345 |

Tabelle 5.1: Durchschnittlicher Payoff (Reward) aus $10^7$ Spielrunden für $s_0$ mit Parameter StopOnRoundOver = true. Gegenüberstellung der Spielzüge Stand, Hit mit zufälligem Rollout und Hit mit optimalem Rollout.

Ausführung des Spielzugs, denn für den Spielzug Stand gibt es nicht die Möglichkeit bis zum Erreichen des Rundenendes suboptimal zu spielen. Für den Parameter StopOnRoundOver = true wird folglich der erwartete Reward des Spielzugs Stand mit „optimaler“ Spielstrategie gegen den erwarteten Reward des Spielzugs Hit mit suboptimaler Spielstrategie gewertet.

Die Tabelle 5.2 zeigt den erwarteten durchschnittlichen Payoff der Spielzüge Hit und Stand mit darauffolgendem zufälligem Rollout bei einer Rolloutdepth von 6 (StopOnRoundOver = false).

| Stand (zufälliges Rollout) | Hit (zufälliges Rollout) |
|---|---|
| −9.802639 | −8.427120 |

Tabelle 5.2: Durchschnittlicher Payoff aus $10^7$ Spielrunden für $s_0$ mit Parameter StopOnRoundOver = false. Gegenüberstellung der Spielzüge Stand und Hit, mit folgendem zufälligem Rollout und einer willkürlich gewählten Rolloutdepth von 6.

Vergleicht man die Ergebnisse aus Tabelle 5.1 und 5.2, so ist zu erkennen, dass durch das auf den Spielzug Stand folgende zufällige Rollout die Differenz in der Bewertung beider Spielzüge aus Tabelle 5.2 sich der Differenz der Bewertungen beider Spielzüge mit optimaler Strategie aus Tabelle 5.1 annähert ($\approx$ 1.4 bis 1.5). Somit konnte für die Variante StopOnRoundOver = false gezeigt werden, dass das unfaire Verhältnis der unterschiedlichen Rollouttiefen abgeschwächt wird. Zum Nachteil dieser Variante ist, dass das Erreichen des Rundenendes innerhalb einer Simulation nicht gewährt ist und somit nicht garantiert ist, dass der Reward für die zuletzt simulierte Runde feststeht. Schlussendlich ist StopOnRoundOver = false nicht zwangsläufig die bessere Variante für alle rundenbasierten Spiele. Sie erzielt lediglich für Blackjack aufgrund des beobachteten Verhaltens bessere Ergebnisse.

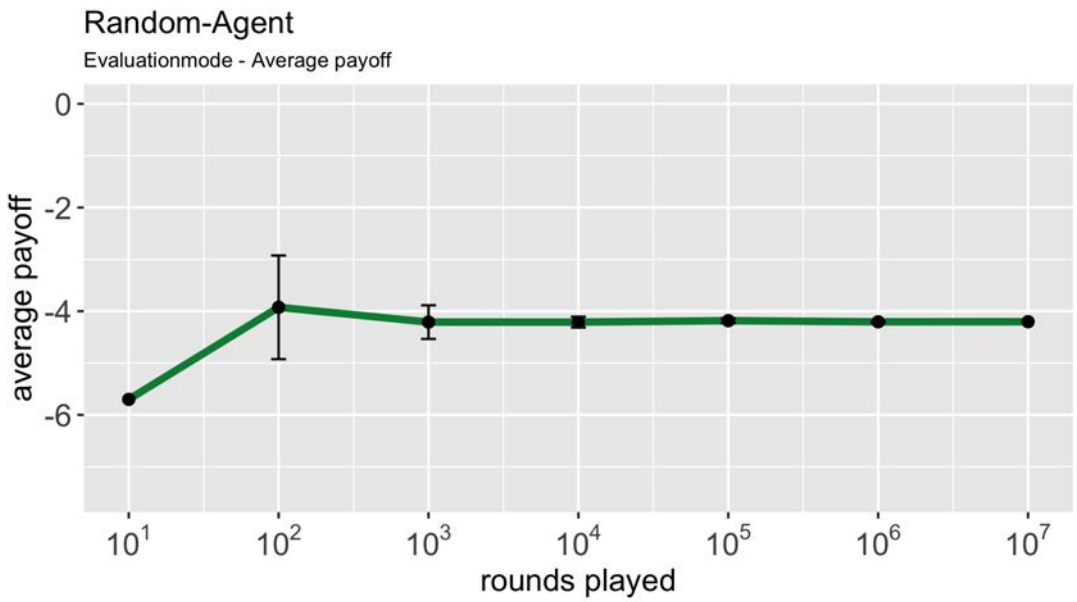


Abbildung 5.1: Random Agent durchschnittlicher Payoff (mit Fehlehrbalken, der Standardabweichung aus zehn Messungen)

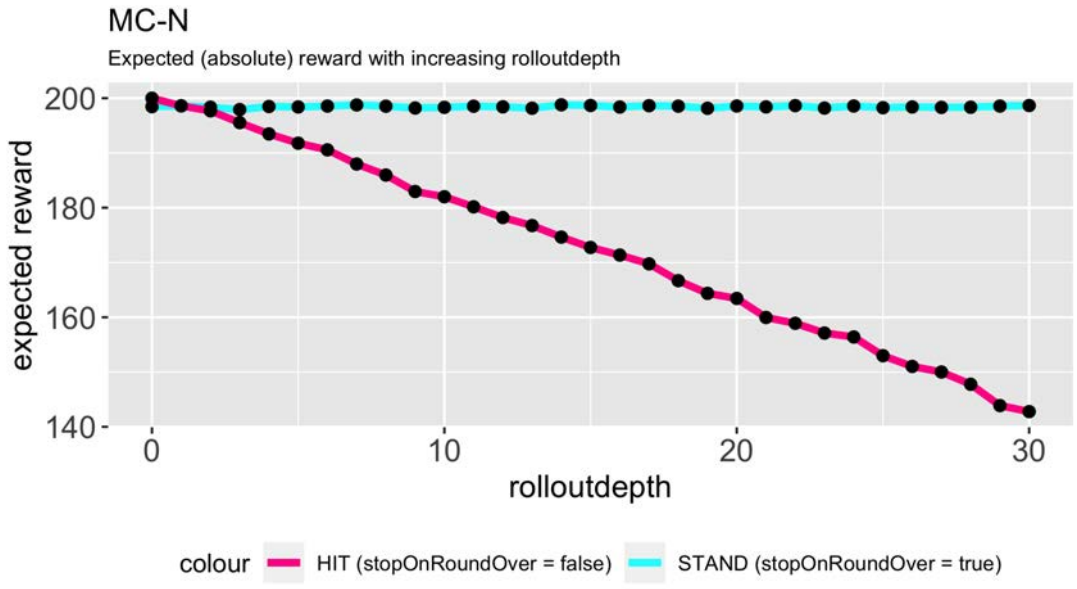


Abbildung 5.2: Extremes Beispiel für erwarteten Reward bei steigender Rolloutdepth

## 5.3 Ausbleibender Lernerfolg von TD-N-Tuple 4 in Blackjack

Beim Debuggen von TD-N-Tuple konnte Wolfgang Konen folgende Beobachtungen für TD-N-Tuple in Blackjack machen. Das Zielsignal einer gegebenen Spielfunktion $V(s)$ und einer Action $a$ ist $r'+\gamma*V(s')$, wobei $r'$ und $s'$ Reward und Game-State nach einem Advance von $s$ mit Action $a$ darstellen. Das Zielsignal selbst in Blackjack schwankt jedoch zu stark, um ein Lernerfolg mit einer Fehlerkorrektur erreichen zu können. Ein erwarteter Reward $r'$ kann nicht den expected Reward einer Action $a$ präsentieren, wenn nicht über alle möglichen $r'$ gemittelt wird. In einem Fall könnte der Spielzug Hit einen Handwert über 21 verursachen, somit wäre hätte der erreichte Game-State einen schlechten Reward, im anderen Fall könnte das Ergebnis genau 21 sein und der erreichte Game-State hätte einen positiven Reward. Das grundlegende Problem ist also, dass TD-N-Tuple Anpassungen der Spielfunktion und Entscheidungen auf Basis von nicht gemittelten expected Rewards vornimmt und trifft. Falls sich vorherige Beobachtungen und Schlussfolgerungen erhärten, ist der TD-N-Tuple Agent für Blackjack ungeeignet. Eine Alternative würde ein Q-Learning oder Sarsa Agent darstellen. Für den Q-Learning Agent lautet die zentrale Zielfunktion

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)] \tag{5.1}$$

Wenn die Lernrate $\alpha$ klein genug gewählt ist, wird für einen vorletzten $Q(s_t, a_t)$ der gemittelte durchschnittliche Reward aller finalen $r_{t+1}$ angenähert, welche von $s_t$ mit Action $a_t$ erreicht werden können.

## 5.4 Ein einfaches Spiel an dem TD-N-Tuple 4 scheitert

Um die für Blackjack gemachten Beobachtungen und Schlussfolgerungen weiter zu prüfen, hat Wolfgang Konen das "Simple Game" (abgeleitet von Blackjack) entwickelt. Im Simple Game erhält der Spieler zu Beginn der Runde eine Karte (mit einem Wert von 1 bis 9). Es gibt die zwei möglichen Spielzüge Hit und Stand.

- Hit: Der Spieler erhält eine weitere Karte (1 bis 9).
- Stand: Beendet die Runde mit der aktuellen Handsumme.

Der Reward ist die Summe aller Karten die der Agent besitzt. Ist die erreichte Summe größer als Neun ist der Reward der Runde Null. Wolfgang Konen konnte berechnen, dass der TD Agent in diesem Spiel mit einer Wahrscheinlichkeit von 30% eine Entscheidung abweichend der optimalen Strategie trifft. [2] Nach Implementation des Simple Games im GBG wurde gemessen, dass der TD Agent in 26% der Fälle einen Zug abweichend der optimalen Strategie gewählt hat. Es wurde außerdem durch Messungen die Annahme bestätigt, dass Q-Learning und Sarsa Agenten die optimale Strategie lernen.

[2] https://github.com/WolfgangKonen/GBG/blob/master/resources/ASimpleGameWhereTDLearningFails.xlsx

## 5.5 Sarsa und Qlearn

Aufgrund des in den Abschnitten 5.3 und 5.4 erkannten Problems (der fehlerhaften Korrektur der Spielfunktion) von TD-N-Tuple wurde für die Agenten Qlearn und Sarsa eine Trainingseinheit in Blackjack umgesetzt. Für die Agenten Qlearn und Sarsa wird im Gegensatz zu TD-N-Tuple ein Lernerfolg erwartet, da die Spielfunktion einen Game State und eine dazugehörige Action bewertet anstatt wie bei TD-N-Tuple nur den Game State. Somit kann der erwartete Reward einer Action zugehörig zu einem spezifischen Game State im Laufe des Lernprozess gemittelt werden. Auf Abbildung 5.3 wird eine Trainingseinheit für den Qlearn Agenten ausgeführt. Anhand des Evaluationsgraphen lässt sich deutlich erkennen, dass es dem Qlearn Agenten möglich ist, die Grundlagen von Blackjack zu erlernen. Unter dieser kurzen Betrachtung erzielt der Agent sogar für die Evaluationsmethoden „Einfache Situationen mit besten Spielzug Hit“(Rot) und „Einfache Situationen mit besten Spielzug Stand“(Blau) ähnlich gute Ergebnisse wie MC und MCTSE. Für den Sarsa Agenten konnten ebenfalls ein Lernerfolg beobachtet werden. Aufgrund der beschränkten Bearbeitungszeit dieser Arbeit kann jedoch der zu beobachtende Lernerfolg nicht weiter untersucht werden.

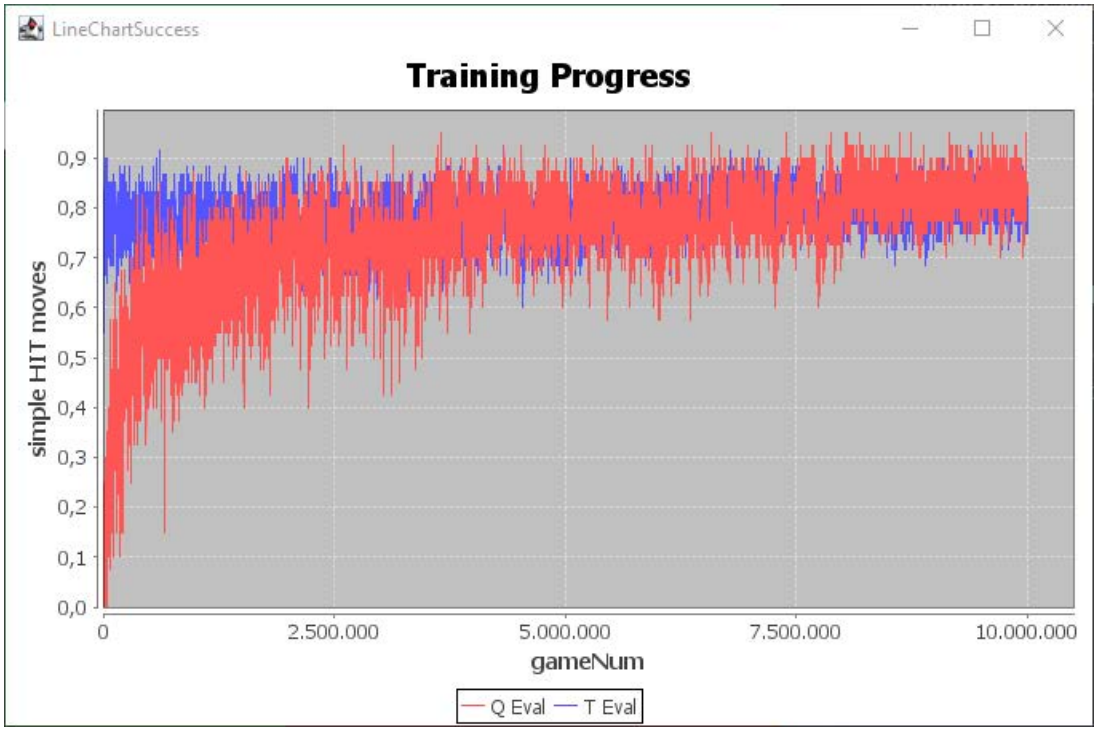


Abbildung 5.3: Trainingseinheit für den Qlearn Agent bestehend aus $10^7$ Spielrunden, mit den Evaluationsmethoden „Einfache Situationen mit besten Spielzug Hit“(Rot) und „Einfache Situationen mit besten Spielzug Stand“(Blau)

# 6 Fazit und Ausblick

Im Vorfeld wurde im Rahmen eines Praxisprojekts (Meißner (2021)) das Game-Learning-Framework General Board Game um eine Blackjack Umgebung erweitert, um eine Basis für die vorliegende Untersuchung zu schaffen. Das stochastische Kartenspiel Blackjack lässt sich als ein rundenbasiertes Spiel imperfekter Informationen mit stark fluktuierenden nichtdeterministischen Ereignissen einstufen. Aufgrund dieser Einstufungen, ist zu vermuten, dass Blackjack eine große Herausforderung für Reinforcement Learning Agenten darstellt.

Im Verlauf dieser Arbeit wurde der Spiel- und Lernerfolg von den allgemeinen Reinforcement Agenten Monte Carlo, Monte Carlo Tree Search Expectimax und Temporal Difference N-Tupel in Blackjack untersucht. Um den Spiel- und Lernerfolg zu messen, wurden Evaluationsstrategien entwickelt, für welche der Basic Strategy Blackjack Agent (BSBJA) implementiert wurde und als Maßstab perfekten Spielens diente. In diesem Zusammenhang konnte im Rahmen eines Tests, bei welchem der durchschnittliche Payoff aller verfügbaren Spielzüge für 20 Spielsituationen gemessen wurde, die für den BSBJA verwendete Basic Strategy validiert werden.

Anhand der Evaluationsstrategien und den daraus resultierenden Messergebnissen lässt sich die anfangs gestellte Forschungsfrage

> „Welche messbaren Spiel- und Lernerfolge können die untersuchten allgemeinen Reinforcement Agenten in Blackjack erzielen?“

beantworten. In der Messung der Spielstärke konnte MC, dicht gefolgt von MCTSE, die besten Ergebnisse erzielen. Beide Agenten konnten einen durchschnittlichen Payoff nahe Null erreichen und erzielen somit deutlich bessere Ergebnisse als ein zufällig spielender Agent, welcher einen durchschnittlichen Payoff von $\approx -4$ erzielt. Die erzielten Ergebnisse von MC und MCTSE nähern sich denen des bestspielenden Agenten BSBJA an. TD-N-Tuple hingegen war es nicht möglich Blackjack besser als ein zufällig spielender Agent zu spielen. Ein Lernerfolg konnte nicht nachgewiesen werden.

Im Kontext der zweiten Forschungsfrage

> „Welchen Einfluss hat die rundenbasierte Natur Blackjacks auf den Spiel- und Lernerfolg der allgemeinen Reinforcement Agent?“

wurde ein Experiment für den Parameter StopOnRoundOver des MC Agenten durchgeführt. In diesem Experiment konnte, entgegen der vorab aufgestellten Prognose, die Erkenntnis gewonnen werden, dass ein Rollout über ein Rundenende hinaus bessere Evaluationsergebnisse für Blackjack erzielt. Für dieses zunächst überraschende Verhalten konnte in einer späteren Analyse die These aufgestellt werden, dass unterschiedlich lange, zufällige Rollouts zu einer unfairen Bewertung der Spielzüge führen, da ein zufälliges Rollout im Mittel einen negativen durchschnittlichen Payoff erzielt und sich somit fortlaufend negativ auf die Bewertung des Spielzugs auswirkt. Diese These konnte

durch Untersuchung einer konkreten Spielsituation bei Betrachtung der Spielzüge Hit und Stand im Zusammenhang mit den darauf folgenden Rollouts erhärtet werden. Das gleiche Verhalten konnte für MCTSE festgestellt werden. Für MC und MCTSE stellt somit die unterschiedliche Rundenlänge, je nach gewähltem Spielzug (bspw. Hit oder Stand), ein Problem dar. Wie bereits im Rahmen der ersten Forschungsfrage diskutiert, konnte für TD-N-Tuple kein Lernerfolg festgestellt werden. Die unterschiedliche Rundenlänge hatte keinerlei Auswirkungen auf den Lernerfolg des Agenten.

Zur Beantwortung der letzten Forschungsfrage

> „Inwiefern stellen die stark fluktuierenden nichtdeterministischen Ereignisse Blackjacks ein Problem für die allgemeinen Reinforcement Agenten in Bezug auf ihren Spiel- und Lernerfolg dar?“

konnte gezeigt werden, dass es aufgrund der stark fluktuierenden nichtdeterministischen Ereignisse dem TD-N-Tupel Agent nicht möglich ist, Blackjack oder ein vereinfachtes Spiel mit ähnlichen Eigenschaften zu lernen. Der ausbleibende Lernerfolg wird durch die vom Agenten vorgenommene Korrektur seiner Spielfunktion, basierend auf dem Ausgang eines einzigen, nicht gemittelten, nichtdeterministischen Ereignisses, verursacht. Andere Temporal Difference Agenten, wie z.B. Q-Learn oder Sarsa, weisen diese Schwachstelle jedoch nicht auf. Dies konnte mittels einer Trainingseinheit der beiden Agenten in der Blackjack Umgebung bestätigt werden, in denen ein Lernerfolg für Black zu erkennen war. Für MC und MCTSE stellen die stark fluktuierenden nichtdeterministischen Ereignisse kein Problem dar, da beide Agenten mittels steigendem Parameter Iterations über zufällige Ereignisse mitteln. Dies ist jedoch für MCTSE mit stark steigender Rechenzeit verbunden.

In Bezug auf die persönlich gestellten Ziele lässt sich sagen, dass die Erweiterung vom GBG Framework um eine Blackjack Umgebung sowie die Untersuchung zu den darin enthaltenen Agenten für Blackjack einen Erfolg darstellen. Die Blackjack Umgebung im GBG stellt nach Implementation eine realistische Version Blackjacks dar und hat nach Überarbeitung der grafischen Oberfläche die persönlich gestellten Anforderungen übertroffen. Die nähere Untersuchung der Agenten hat schwerwiegende Probleme und Schwierigkeiten aufgedeckt (z.B. die Schwächen des Temporal Difference Agenten im spezifischen Anwendungsfall Blackjack), welche teilweise trotz großer Bemühungen nicht gelöst werden konnten, aber dennoch interessante Erkenntnisse in Bezug auf die Funktionsweise der Agenten und die Spielumgebung geliefert haben.

Es lässt sich resümieren, dass nur bei den Agenten MC und MCTSE ein messbarer Spielerfolg nachgewiesen werden konnte. Dennoch weisen beide Agenten bei rundenbasierten, nichtdeterministischen Spielen mit imperfekten Informationen durchaus Limitierungen auf, weshalb es sinnvoll erscheint den Spiel- und Lernerfolg weiterer Agenten, wie z.B. Q-Learn oder Sarsa, für Blackjack zu untersuchen. Außerdem könnten die vorhanden Evaluationsstrategien um Strategien erweitert werden, welche speziell die Spielzüge Double Down, Surrender und Split im Einzelnen betrachten. Somit könnte die Zusammensetzung von Evaluationsergebnissen allgemeiner Spielstärke besser analysiert und mögliche Schwächen oder Stärken eines Agenten isoliert werden.

Schlussendlich könnten Spiele untersucht werden, welche ähnliche Eigenschaften wie Blackjack besitzen, um die für Blackjack gewonnen Erkenntnisse auf ähnliche Spiele zu übertragen und somit die gewonnen Erkenntnisse weiter zu generalisieren.

# Abbildungsverzeichnis

# Tabellenverzeichnis

# Literaturverzeichnis


[Aigner u. Ziegler 2010] In: Aigner, Martin ; Ziegler, Günter M.: *Buffon's needle problem*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2010. – ISBN 978–3–642–00856–6, 155–158

[Andrieu u. a. 2003] Andrieu, Christophe ; Freitas, Nando de ; Doucet, Arnaud ; Jordan, Michael I.: An Introduction to MCMC for Machine Learning. In: *Machine Learning* 50 (2003), Jan, Nr. 1, 5-43. http://dx.doi.org/10.1023/A:1020281327116. – DOI 10.1023/A:1020281327116. – ISSN 1573–0565

[Baldwin u. a. 1956] Baldwin, Roger R. ; Cantey, Wilbert E. ; Maisel, Herbert ; McDermott, James P.: The Optimum Strategy in Blackjack. In: *Journal of the American Statistical Association* 51 (1956), Nr. 275, 429–439. http://www.jstor.org/stable/2281431. – ISSN 01621459

[Brockman u. a. 2016] Brockman, Greg ; Cheung, Vicki ; Pettersson, Ludwig ; Schneider, Jonas ; Schulman, John ; Tang, Jie ; Zaremba, Wojciech: OpenAI Gym. In: *CoRR* abs/1606.01540 (2016). http://arxiv.org/abs/1606.01540

[Chaslot 2010] Chaslot, Guillaume Maurice Jean-Bernard C.: *Monte-Carlo tree search*, Diss., 2010

[De Granville ] De Granville, Charles: Applying Reinforcement Learning to Blackjack Using Q-Learning. In: *University of Oklahoma.*

[Konen 2015] Konen, Wolfgang: Reinforcement Learning for Board Games: The Temporal Difference Algorithm / Research Center CIOP (Computational Intelligence, Optimization and Data Mining). Version: 2015. http://dx.doi.org/10.13140/RG.2.1.1965.2329. Cologne University of Applied Sciences, 2015. – Forschungsbericht

[Konen 2019] Konen, Wolfgang: General Board Game Playing for Education and Research in Generic AI Game Learning. In: Perez, Diego (Hrsg.) ; Mostaghim, Sanaz (Hrsg.) ; Lucas, Simon (Hrsg.): *IEEE Conference on Games*. London, 2019

[Konen 2020] Konen, Wolfgang: The GBG Class Interface Tutorial V2.2: General Board Game Playing and Learning / Research Center CIOP (Computational Intelligence, Optimization and Data Mining), Oct. Version: 2020. http://www.gm.fh-koeln.de/ciopwebpub/Konen20d.d/TR-GBG.pdf. 2020. – Forschungsbericht

[Konen u. Bartz-Beielstein 2008] Konen, Wolfgang ; Bartz-Beielstein, Thomas: Reinforcement Learning: Insights from Interesting Failures in Parameter Selection. In: Rudolph, Günter (Hrsg.) u. a.: *PPSN'2008: 10th International Conference on Parallel Problem Solving From Nature, Dortmund*. Berlin : Springer, 2008, 478–487

[Kutsch 2017] Kutsch, Johannes: *KI-Agenten fur das Spiel 2048: Untersuchung von Lernalgorithmen für nichtdeterministische Spiele*. http://www.gm.fh-koeln.de/~konen/research/PaperPDF/BA_JohannesKutsch_Final-2017.pdf. Version: Jan 2017. – Bachelor thesis

[Lindner u. York 2017] Lindner, Roland ; York, New: Zukunft der Menschheit: Künstliche Intelligenz überall. In: *FAZ.NET* (2017). https://www.faz.net/1.5130736. – ISSN 0174–4909

[Meißner 2021] Meissner, Simon: *Entwicklung eines Black Jack Clienten in der Game Learning Umgebung General Board Game*. Th-Köln, Jan 2021. – Praxisprojekt

[Shonkwiler u. Mendivil 2009] In: Shonkwiler, Ronald W. ; Mendivil, Franklin: *Introduction to Monte Carlo Methods*. New York, NY : Springer New York, 2009. – ISBN 978–0–387–87837–9, 1–49

[Silver u. Schrittwieser 2017] Silver, David ; Schrittwieser, et a.: Mastering the game of Go without human knowledge. In: *Nature* vol. 550 (2017), Oktober, Nr. no. 7676, pp. 354–359. http://dx.doi.org/10.1038/nature24270. – DOI 10.1038/nature24270. – ISSN 0028–0836, 1476–4687

[Sutton u. Barto 2018] Sutton, Richard S. ; Barto, Andrew G.: *Reinforcement Learning: An Introduction*. Second Edition. The MIT Press, 2018 http://incompleteideas.net/book/the-book-2nd.html

[Świechowski u. a. 2021] Świechowski, Maciej ; Godlewski, Konrad ; Sawicki, Bartosz ; Mańdziuk, Jacek: *Monte Carlo Tree Search: A Review of Recent Modifications and Applications*. Eingereicht an das AI Review journal am 8 Mar 2021. https://arxiv.org/abs/2103.04931. Version: Jan 2021

[Tesauro 1994] Tesauro, Gerald: TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play. In: *Neural Computation* 6 (1994), 03, Nr. 2, 215-219. http://dx.doi.org/10.1162/neco.1994.6.2.215. – DOI 10.1162/neco.1994.6.2.215. – ISSN 0899–7667

[Tromp 2016] Tromp, John: The number of legal Go positions. In: *International Conference on Computers and Games* Springer, 2016, S. 183–190

[Zha u. a. 2019] Zha, Daochen ; Lai, Kwei-Herng ; Cao, Yuanpu ; Huang, Songyi ; Wei, Ruzhe ; Guo, Junyu ; Hu, Xia: RLCard: A Toolkit for Reinforcement Learning in Card Games. In: *CoRR* abs/1910.04376 (2019). http://arxiv.org/abs/1910.04376

Simon Meißner

# Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben.

Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Gummersbach, 27. April 2021 

Simon Meißner