Abbildung von Drei-Spieler-Situation in Zwei-Spieler-Situation im Texas-Hold'em-No-Limit-Poker

Mapping of Three-Player Situations into Two-Player Situations in Texas-Hold'em-No-Limit Poker

Bachelor-Thesis von Timm Lilienthal aus Groß Umstadt

Tag der Einreichung:

1. Gutachten: Prof. Dr. Johannes Fürnkranz
2. Gutachten: Dr. Eneldo Loza Mencía

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
Knowledge Engineering Group

Abbildung von Drei-Spieler-Situation in Zwei-Spieler-Situation im Texas-Hold'em-No-Limit-Poker
Mapping of Three-Player Situations into Two-Player Situations in Texas-Hold'em-No-Limit Poker

Vorgelegte Bachelor-Thesis von Timm Lilienthal aus Groß Umstadt

1. Gutachten: Prof. Dr. Johannes Fürnkranz
2. Gutachten: Dr. Eneldo Loza Mencía

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 2. Januar 2019

(T.Lilienthal)

Inhaltsverzeichnis

1 Einleitung 3

2 Poker 4
2.1 Pokerbegriffe . . . 4
2.2 No-Limit-Texas-Hold'em Poker . . . 5
2.3 Regeln in der Arbeit . . . 8

3 Verwandte Arbeiten 9
3.1 TU-Darmstadt Poker Framework . . . 9
3.2 CFR No-Limit-Texas-Hold'em Bot . . . 9
3.3 Counterfactual Regret Minimization . . . 10
3.4 Andere Arbeiten . . . 10

4 Mapping der Gamestates 12
4.1 Anpassung der Spieler . . . 12
4.2 Anpassung der Aktionshistories . . . 13
4.2.1 Look-Ahead Tabel . . . 13
4.2.2 Mapping Algorithmus . . . 15
4.2.3 Anpassung der Pot Size . . . 16
4.2.4 Hero Call Problem . . . 17
4.3 Zusammenfassend . . . 20

5 Testaufbau 21

6 Ergebnisse 23
6.1 Gewinn/Verlust . . . 23
6.2 Gewinn/Verlust Verteilung . . . 27
6.3 Aktionshäufigkeiten . . . 31
6.4 Aktionsvariation . . . 31
6.5 Verluste durch das Mapping . . . 32

7 Fazit 34

Literaturverzeichnis 35

A Quellcode 36

1 Einleitung

Spiele sind eine gute Möglichkeit um Problemstellungen für KIs zu finden die es zu lösen gilt. Hierbei geht es weniger darum die Spiele zu lösen, sondern Programmen beizubringen mehr oder weniger komplexe Probleme anzugehen und diese mit Blick auf moderne Möglichkeiten zu lösen. So hilft es nicht einen Agenten zu erstellen, der ein Problem zwar lösen kann, aber unrealistisch hohe Ressourcen Anforderungen besitzt. Ein gutes Beispiel für dieses Problem wäre hier Schach, so ist es nicht besonders schwierig einen Agenten zu schreiben, der für jede Spielsituation den bestmöglichen Zug findet, jedoch wären die benötigten Ressourcen so hoch, das kein moderner Rechner diese Agenten in einem angemessen Zeitrahmen ausführen könnte.

In dieser Arbeit geht es um das Spiel No-Limit-Texas-Hold'em Poker. Im Gegensatz zu Schach hat Pokern jedoch einige Eigenarten, die das Spiel um einiges komplexer machen. Ein paar davon sind:

- Partially Observerble:
  In einem Schachspiel sind zur jeder Zeit des Spiels alle Informationen verfügbar. In einem Pokerspiel ist das nicht der Fall. So kenne ich die Handkarten des Gegners nicht. Auch werden im Fall von Texas-Hold'em Karten erst im Laufe des Spiels bekannt gegeben. Durch diese Faktoren ändert sich die Komplexität des Spiels stark, da davon ausgegangen werden muss, dass mein Gegner jede Kartenkombination auf der Hand haben könnte und im Laufe des Spiels jede mögliche Karte aufgedeckt wird.

- Glücksfaktor:
  Da die Verteilung der Karten zufällig geschieht unterliegt Pokern einem gewissen Glücksfaktor. So ist es möglich die richtige Entscheidung zu treffen, aber trotzdem zu verlieren, was je nach Spielmodus das Ergebnis verändern kann. Diese Faktoren müssen beim Erstellen einer Strategie bedacht werden.

- Vergangene Aktionen:
  In Spielen wie Schach wähle ich eine Aktion aus und kann diese im nächsten Schritt vergessen haben, da sie keinen weiteren Einfluss mehr auf das Spiel nimmt. Beim Pokern ist dieses jedoch nicht möglich, da einige Informationen wichtig für den weiteren Verlauf des Spieles sind. Auch hiermit erhöht sich die Komplexität des Spiels.

- Multiplayer:
  Ein weiterer Faktor, der die Komplexität eines Pokerspiels stark erhöht, ist die Möglichkeit mit mehr als zwei Spielern zu spielen. Texas-Hold'em wird mit bis zu 10 spielen gespielt, dies sorgt dafür, dass auf eine Aktion des Agenten bis zu 9 anderen Aktionen folgen können.

Die bisherige Forschung im Gebiet des Texas-Hold'em Poker, beschränkt sich vor allem auf Heads-up Poker, sprich Zwei-Spieler-Poker. Heads-Up-Texas-Hold'em-Limit gilt als gelöst [14]. In Heads-Up-Texas-Hold'em-No-Limit gibt es erste Bots die Profispieler schlagen können [10]. Anders sieht es im Gebiet der Multiplayer Texas-Hold'em Spiele aus. Hier gibt es bisher deutlich weniger und qualitativ schlechtere Ansätze, wie z.B. [12] und [5].

In dieser Arbeit wird nun ein Ansatz entwickelt und getestet um einen Drei-Spieler-Texas-Hold'em-No-Limit Agenten zu entwickeln. Dabei werden bereits existierende Heads-Up Bots verwendet [11]. Probleme ergeben sich hier beim Bewerten des Agenten, da es keine guten Agenten gibt, mit dem Bot verglichen werden können.

2 Poker

Poker ist ein Kartenspiel, das meistens mit einem Blatt mit 52 Karten gespielt wird. Hierbei versucht ein Spieler mit 5 Karten die beste Hand zu bilden, um eine Runde für sich zu entscheiden. Dabei setzen die Spieler einen Einsatz und "wetten" auf den Gewinn ihres Blattes ohne das genaue Blatt ihrer Gegenspieler zu kennen. Hierbei geht es nicht darum möglichst viel Hände zu gewinnen, sondern die größtmögliche Anzahl an Chips zu gewinnen.

Poker gibt es in verschiedenen Varianten. Die wohl Bekannteste ist No-Limit-Texas-Hold'em, welche auch die hier in der Arbeit verwendete Spielvariante ist. Vor allem bei Pokerbots ist Limit-Texas-Hold'em sehr beliebt, eine Variante des Pokerspiels, bei dem die Höhe der Einsätze festgelegt sind. [9]

2.1 Pokerbegriffe

Hier eine kurze Übersicht von relevanten Pokerbegriffen [9]:

- Call:
  Der Call ist eine Aktion im Pokern, bei dem der Spieler den aktuell höchsten Einsatz mitgeht und erhöht so die Potsize. In dieser Arbeit wird diese Aktion als "C" dargestellt.

- Check:
  Ein Check ist eine Aktion im Pokern. Sie ist sehr ähnlich wie ein Call, tritt jedoch nur dann auf, wenn keine Pot-Erhöhung geschehen ist und ein Spieler ohne einen höheren Einsatz zu tätigen weiter spielen möchte. In dieser Arbeit wird die Aktion ebenfalls als "C" um den Wert 0 modelliert.

- Raise:
  Ein Raise ist eine Aktion im Pokern. Bei dieser wird der aktuelle Einsatz erhöht, die Höhe ist beliebig, muss aber mindestens um einen Big Blind höher sein als der aktuell höchste Einsatz und kleiner gleich der Stacksize der Spielers. In dieser Arbeit wird die Aktion als "Rx" wobei x eine natürliche Zahl größer gleich das doppelte des Big Blinds sein muss.

- Fold:
  Ein Fold ist eine Aktion im Poker. Bei dieser gibt der Spieler seine aktuelle Hand auf und nimmt nicht weiter an der Runde teil. Alle seine bisher gesetzten Chips bleiben jedoch im Pott. Gefolded kann nur dann werden, wenn der Amonut-to-Call höher als 0 ist, was bedeutet, dass es bereits eine Pot-Erhöhung in dieser Street gegeben haben muss. In dieser Arbeit wird diese Aktion als "F" dargestellt.

- All-in:
  Ein All-in beschreibt eine Aktion im Poker, bei der alle verbleibenden Chips eines Spielers eingesetzt werden. Es ist der höchste Raise den ein Spieler tätigen kann. In dieser Arbeit wird diese Aktion mit einem "R20000" modelliert.

- Pot/ Potsize:
  Die Potsize oder der Pot beschreiben den Wert der aktuellen Einsätze aller Spieler und den Gewinn der aktuellen Runde.

- Stack:
  Der Stack beschreibt die Chips, die einem Spieler zu Verfügung stehen. Die Stacksize beschreibt den Wert dieser Chips.

- Small Blind:
  Der Small Blind beschreibt den Einsatz den die Position des SB, ebenfalls Small Blind genannt, zu Beginn einer Runde tätigen muss.

- Big Blind:
  Der Big Blind beschreibt den Einsatz den die Position des BB, ebenfalls Big Blind genannt, zu Beginn einer Runde tätigen muss.

- Street:
  Die Street beschreibt eine der vier verschiedenen Setz Runden.

- preFlop:
  Der PreFlop ist die erste Setzrunde, hier werden die Blinds gesetzt und es sind noch keine Community Cards bekannt.

- Flop:
  Der Flop ist die zweite Setzrunde, hier werden 3 Community Cards bekannt gegeben.

- Turn:
  Der Turn ist die dritte Setzrunde, hier wird eine weitere Community Card aufgedeckt.

- River:
  Der River ist die vierte Setzrunde, hier wird eine weitere Community Card bekannt gegeben. Dies ist die letzte Street und wenn am Ende der Street noch mehr als ein Spieler aktiv an der Hand teilnimmt kommt es zum Showdown.

- Community Cards:
  Die Community Cards sind die öffentlichen Karten die jedem Spieler bekannt sind und von diesem benutzt werden können, um die Hand des Spielers zu bilden.

- Showdown:
  Beim Showdown werden die geheimen Handkarten aufgedeckt und allen Mitspielern gezeigt. Ein Showdown tritt entweder am Ende des Rivers auf, wenn noch mindestens zwei Spieler aktiv an der Hand teilnehmen oder wenn alle aktiven Spieler All-in gegangen sind.

- Hand:
  Die Hand eines Spielers beschreibt die fünf Karten, die ein Spieler benutzt, um im Fall eines Showdowns seine Hand bewerten zu können. Sie setzt sich aus den beiden geheimen Handkarten und den fünf Community Cards zusammen, wobei zwei dieser Karten nicht ausgewählt werden.

- Amount-to-Call:
  Der Amount-to-Call beschreibt die Differenz zwischen dem Einsatz des Spielers und dem höchsten bisher getätigten Einsatz. Dieser ist interessant, da es den minimalen Einsatz darstellt, den ein Spieler tätigen muss, um weiterhin an der Hand teilzunehmen und welcher im Fall eines Calls von dem Stack des Spielers abgezogen wird.

- Pot Odds:
  Die Pot Odds beschreiben das Verhältnis des Amount-to-Call zu der Potsize. Je geringer dieser Wert desto geringer ist der zu tätigende Einsatz im Blick auf den möglichen Gewinn. Verrechnet man diesen Wert mit der Gewinnwahrscheinlichkeit seiner Hand, kann eine Strategie berechnet werden. [13]

- All-in Equity:
  Die All-in Equity beschreibt die Wahrscheinlichkeit eine Hand zu gewinnen, wenn alle aktiven Spieler zu dem Zeitpunkt All-in wären. So beträgt z.nbsp;B. die All-in Equity der Hand AdAc Preflop in einem 2-Spieler-Spiel 84,9 [3] % gegen eine zufällige Hand des Gegners.

- Hero:
  Der Hero beschreibt den gerade betrachten Spieler. Hero Aktionen sind Aktionen, die von diesem Spieler getätigt wurden. Ist der Hero am Zug, so wird auf die Aktion dieses Spielers gewartet.

2.2 No-Limit-Texas-Hold'em Poker

Beim Texas-Hold'em Poker [9] bekommt jeder Spieler zwei Handkarten, die nur dem jeweiligen Spieler bekannt sind. Dazu gibt es noch fünf Community Cards, die von allen Spielern genutzt werden können. Diese werden zu verschiedenen Zeitpunkten im Spiel aufgedeckt. Die Karten besitzen 13 verschiedene Kartenwerte und 4 verschiedene Kartenfarben.

$$Kartenwert : W = \{2, 3, 4, 5, 6, 7, 8, 9, T, J, D, K, A\}$$

$$Farbe : F = \{\clubsuit, \diamondsuit, \heartsuit, \spadesuit\}$$

$$Karten : K = \{W \times F\}$$

Die Farben können jedoch auch als {c,d,h,s} dargestellt werden. Die Anzahl der mitspielenden Spieler kann variieren. Theoretisch ist es möglich das Spiel mit bis zu 23 Spielern zu spielen, in der Realität wird es aber meistens mit maximal zehn Spielern an einem Tisch gespielt. Da sich diese Arbeit mit Drei-Spieler und Zwei-Spieler Pokern beschäftigt, wird sich auf den Ablauf dieser beiden konzentriert.

Am Anfang des Spiels werden Positionen vergeben. In Zwei-Spieler-Spielen gibt es die Positionen Small Blind und Big Blind. In einem Drei-Spieler-Spiel kommt noch die dritte Position des Dealers (Button) hinzu.

$$Positionen2Spieler := \{SB, BB\}$$

$$Positionen3Spieler := \{BU, SB, BB\}$$

In der ersten Runde wird die Position des SB zufällig verteilt. Dem nächsten Spieler im Uhrzeiger vom SB wird der BB zugewiesen, in einem Drei-Spieler-Spiel wird dem dritten Spieler die Position des BU zugewiesen. Zu Beginn jeder Runde muss der SB einen festen Betrag setzen, ebenfalls Small Blind genannt wird, welcher dem Pot hinzugefügt wird. Der BB muss das Doppelte vom Small Blind zum Pot hinzufügen, Big Blind genannt. Nun bekommen alle Spieler zwei Karten ausgeteilt, und die erste Setzrunde beginnt. Alle Spieler können zwischen verschieden Aktionstypen wählen. Möglich ist ein Call(c), ein Raise(r) und ein Fold(f).

$$Aktion := \{c, rMin, ..., rMax, f\}$$

Wobei min dem Big Blind entspricht und max der Stacksize eines Spielers, dazwischen ist jeder ganzzahlige Wert möglich.

- Bei einem Call wird der Einsatz des Spielers auf das bisher höchste Gebot erhöht, für den Fall, dass keine Erhöhung stattgefunden hat, nennt man den Call einen Check.
- Bei einem Raise wird der Einsatz um einen bestimmten Wert erhöht. Dieser Betrag ist frei wählbar, muss jedoch mindestens einen Big Blind höher sein als der aktuell höchste Einsatz und kann den Wert des eigenen Stacks nicht überschreiten.
- Bei einem Fold werden die Handkarten weggeworfen und der Spieler nimmt nicht weiter an der aktuellen Setzrunde teil, verliert aber seien bisher getätigten Einsatz. Bei einem Heads-Up Spiel ist die Runde nach einem Fold beendet.

Für den Fall, dass alle aktiven Spieler den gleichen Einsatz gebracht haben, wird die aktuelle Setzrunde beendet und die nächste Setzrunde eingeleitet. Alle Setzrunden, auch Streets genannt, haben verschiedenen Namen.

$$Street := \{PreFlop, Flop, Turn, River\}$$

- Der PreFlop ist die erste Street, hier sind noch keine Community Cards verfügbar, so das nur mit verdeckten Handkarten gespielt wird.
- Zu Beginn des Flops werden 3 Community Cards aufgedeckt.
- Im Turn wird eine weitere Community Card aufgedeckt.
- Im River wird die letzte Community Card aufgedeckt. Wenn am Ende dieser Street noch mehr als ein aktiver Spieler am Spiel teilnimmt, müssen alle noch aktiven Spieler ihre Karten aufdecken und die Hände werden ausgewertet.

Ein Spieler gewinnt den gesamten Pot, wenn kein anderer Spieler mehr aktiv am Spiel teilnimmt oder wenn er bei einem Showdown die bestbewertete Hand hält. Sollten die Hände von mehreren Spielern gleich stark bewertet werden, wird der Pot unter diesen Spielern aufgeteilt. Die Hände werden nach dem Schema in Tabelle 1 bewertet, wobei die stärkste Hand oben steht und die schwächste Hand unten steht.

Tabelle 1: Hand Wertigkeit in Texas-Hold'em

Bezeichnung	Beschreibung	Beispiel	Gleichstand
Royal Flush	Straße von Ass bis zehn derselben Farbe	A♣K♣Q♣J♣T♣	-
Straight Flush	Straße der Selben Farbe	4♣5♣6♣7♣8♣	Höchster Karten Wert
Vierling	Vier Karten desselben Wertes	4♣4♢4♡4♠K♡	Höherer Wert des Vierlings, bei Gleichstand höhe der fünften Karte
Full House	Drilling des Wertes A und Paares des Werts B	A♣A♠A♡K♡K♣	Höherer Wert des Drillings, bei Gleichstand Höherer Wert des Paares
Flush	Fünf Karten derselben Farbe	A♣T♣6♣2♣9♣	Höchster Karten Wert
Street	Fünf anneinander gereite Karten	4♢5♠6♣7♡8♠	Höchster Karten Wert
Drilling	Drei karten desselben Wertes	4♢4♠4♣8♡K♠	Höherer Wert des Drillings, bei Gleichstand Wert der fünften Karte
Zwei Paar	Zwei Paare	4♢4♠8♣8♡K♠	Wert des höhren Paares, bei Gleichstand Wert des niedrigern Paares, bei Gleichstand Wert der fünften Karte
Paar	Zwei Karten desselben Wertes	4♢4♠8♣T♡K♠	Wert des Paares, bei Gleichstand Wert der hösten Karte
High Card	Die Höchste Karte	A♢4♠8♣T♡K♠	Wert der Höchsten Karte

Nachdem eine Runde beendet ist, werden die Positionen im Uhrzeigersinn weitergegeben und die nächste Runde beginnt. Das Ende eines Spiels unterscheidet sich, je nach gewähltem Spielformat. Bei Turnieren scheidet ein Spieler aus wenn dieser keine Chips mehr zur Verfügung hat. Das Spiel endet, wenn nur noch ein Spieler Chips besitzt. Ein weiterees Format sind Cash Games hier können Spieler zum Start einer Runde einsteigen und jederzeit wieder aussteigen. Die Einzahlung die ein Spieler dabei beim Einstieg in die Runde tätigen muss und sein Stack bildet kann hierbei Variieren. Bei einem Cash Game bleiben die Blinds über das gesamte Spiel immer gleich.

Unterschiede zwischen einem Zwei-Spieler-Spiel und einem Zwei-Spieler-Spiel bestehen vor allem in den Positionen und den auftretenden Aktionsabläufen. Der größter Unterschied zwischen den beiden Spielvarianten ist die zusätzliche Position. Diese sorgt für einen anderen Ablauf in den Streets, so agiert der SB in einer Zwei-Spieler-Situation Preflop immer zuerst. In einer Drei-Spieler-Preflop-Situation beginnt jedoch der BU. Post-Preflop beginnt sowohl in einem Zwei-Spieler-Spiel als auch in einem Drei- Spieler-Spiel immer der SB. Wobei Post-PreFlop alle Streets nach dem PreFlop beschreibt. Dies sorgt für folgende Setzreihenfolge:

Tabelle 2: Setzt Reihenfolge

	Zwei-Spieler	Drei-Spieler
Preflop	SB:BB	BU:SB:BB
Flop	SB:BB	SB:BB:BU
Turn	SB:BB	SB:BB:BU
River	SB:BB	SB:BB:BU

Ein weiterer Unterschied ergibt sich bei den möglichen Spielableufen. Ein Spielablauf beschreibt die Abfolge der Aktionen in einer Street. In einem Zwei-Spieler-Spiel endet eine Street immer mit dem ersten Call, außer dieser ist die erste Aktion in der Street. In einer Drei-Spieler-Situation können jedoch mehrer Calls in einem Aktionsablauf vorkommen, ohne dass die Street benndet sein muss. Erst wenn zwei Calls aufeinanderfolgen ist die Street in einem Drei-Spieler-Spiel in jedem Fall beendet, außer sie kommen zu Beginn einer Street.

Ein Beispiel für den Ablauf einer Drei-Spieler Street wäre :

, wobei ein “R” einen Raise auf den Einsatz des Stacks des ausgeführten Spielers um den nachfolgenden Wert symbolisiert. “C” symbolisiert einen Call und “F” symbolisiert einen Fold. Gleichfarbige Aktionen wurden vom selben Spieler getätigt. Der Zwei-Spieler-Aktionsablauf wäre bereits nach dem zweiten Call beendet.

2.3 Regeln in der Arbeit

Für diese Arbeit werden folgende Werte verwendet.

- Small Blind: 50
- Big Blind: 100
- Stack Size: 20000

Das Format ist eine leicht abgewandelte Cash Game Form, in der die Anzahl der Runden festgelegt ist. Des Weiteren wird mit automatischem Refill und Pay-out gespielt. Beim automatischen Refill werden Verluste von Chips nach jeder Runde wieder aufgefüllt, bei automatischem Pay-out werden Gewinne sofort ausgezahlt. Dadurch ist die Ausgangslage vor jeder Runde für jeden Spieler gleich, egal wie dieser die Runde zuvor abgeschnitten hat. Diese Regeln wurden in der ACPC Poker Challenge [1] verwendet.Bei der ACPC Poker Challenge handelt es sich um einen Wettbewerb, in dem Bots in verschiedenen Kategorien gegeneinander antreten. Einer in dieser Arbeit verwendete Bot wurden für diese Challenge entwickelt, was das Verwenden dieser Spielregeln voraussetzt, um diesen verwenden zu können.

3 Verwandte Arbeiten

3.1 TU-Darmstadt Poker Framework

Die Arbeit wurde mit dem TU-Darmstadt Poker Framework entwickelt und in dieses integriert [4]. Dieses Framework bietet die Möglichkeit, Bots an einen Server anzumelden und diese gegeneinander Spielen zu lassen. Das Format, in dem die Zustände innerhalb des Frameworks übergeben werden, sind passend zu dem der ACPC Challenge [1]. Zum Testen von Bots gibt es zusätzlich noch einige Bots, die bereits in dem Framework integriert wurden und auch zum Testen verwendet werden.

- EVBot:
  Der EVBot im Poker Framework, versucht über die All-in Equity wenn alle Spieler All-in gehen würden und die Pot Odds miteinander zu verrechnen um so eine passende Aktion zu wählen. Dies sorgt für ein sehr aggressives Spiel des Bots.

- MinRaiseBot:
  Der MinRaiseBot erhöht immer, wenn es ihm möglich ist um das minimum, in diesem Fall jeweils um 100. Dies tut er unabhängig von seiner Handstärke und beschreibt daher einen sehr schwachen Bot.

- AllinBot:
  Bei einem All-in Bot ist jede Aktion des Bots ein All-in, in diesem Fall ein Raise von 20000. Dies tut er unabhängig von der Handstärke und beschreibt so einen sehr schwach spielenden Bot.

3.2 CFR No-Limit-Texas-Hold'em Bot

Dieser Abschnitt beschreibt die Arbeit [11].
Die in der Arbeit erzeugten Mappings, sollten von einem starken Heads-up Agenten interpretiert werden. Viele moderne Pokerbots basieren auf dem Counterfactual Regret Minimization Algorithmus, einem Algorithmus zum Erstellen einer Nash Equilibrium in einem Nullsummenspiel (mehr in 3.3). Im in 3.1 erwähnten Poker Framework existiert bereits ein CFR Heads-Up-Texas-Hold'em-No-Limit und Limit Bot. Hier wird jedoch nur auf die No-Limit-Variante eingegangen.

Der oben beschriebene Bot hat bereits mit dem CFR Algorithmus eine Strategie gelernt, die jederzeit abrufbar ist. Da Heads-Up-Texas-Hold'em-No-Limit ungefähr 10^{160} verschiedene Spielsituationen hat, [10] moderne Systeme jedoch aktuell nur Spiele mit maximal ca 10^{14} [10]Situationen berechnen kann, müssen die verschiedenen Aktionen stark heruntergebrochen werden. Hierbei werden mehrere Abstraktionen unternommen um die Verkleinerung des Situationsraumes möglich zu machen.

- Aktionen-Abstraktion:

  Da beim No-Limit Pokern, jeder mögliche Raise erlaubt ist, solange er größer gleich des Big Blinds und kleiner gleich der Stacksize ist, müssten der Raise 1500 und Raise 1501 zwei verschiedene Aktionen berücksichtigt werden. Da hier jedoch die Raises sehr ähnlich sind, können sie zu einem Raise zusammengefasst werden. So benutzt der CFR Bot zusätzlich zu den Aktionen Fold und Call/Check jeweils Raises in verschiedenen Höhen, abhängig von der Potsize. Die Höhen unterscheiden sich zwischen Initialeinsätzen, die jeweils als Erstes in einer Street getätigt werden, und darauf folgende Einsatzerhöhungen. Folgende Setzgrößen werden vom Bot verwendet, wobei der Wert immer das x Fache der Potsize beschreibt:
  Initial: 0.25, 0.5, 0.75, 1, 1.5, 2, 4, 8, 15, 25, 50.
  sonstige Einsatzerhöhung: 0.25, 0.5, 1, 2, 4, 8, 15, 25, 50
  Des Weiteren ist immer ein All-in möglich. Sollte einer dieser Einsätze Preflop höher als 60 Prozent der Stacksize sein, wird dieser automatisch auf All-in abgebildet. Ab dem Flop erhöht sich dieser Prozentsatz bis ein Raise als All-in gewertet wird auf 75%.

- Karten-Abstraktion:

  Bei den Kartenabstraktionen wird versucht Handkarten, die gleich gespielt werden, in ein Bucket zu verschieben und diese ab jetzt gleichzubehandeln. Ein Einfaches Bucketing wäre eine reine Einteilung in isomorphe Hände. Dies sind Hände, bei denen nur die Farben der Karten unterschiedlich sind, sie sonst aber identisch sind. Ein Beispiel hierfür wäre AcKc und AdKd . Beide Hände können Preflop gleichbehandelt werden, da die All-in Equity für beide Hände gleich ist und bevor Community Cards aufgedeckt werden auch die selbe Taktik verfolgen. Die Karten müssen jedoch noch weiter in Buckets eingeteilt werden, hierbei kommt es auch zu Informationsverlusten [7]. Es werden einmal die Bordkarten und einmal die Handkarten in Buckets eingeteilt und diese dann miteinander kombiniert, sodass jede mögliche Kombination zwischen Hand- und Bordkarten je einem Bucket zugeordnet wurde.

Um das Spiel weiter zu vereinfachen, kann beim Bucketing zwischen Perfect Recall und Imperfect Recall unterschieden werden. Bei einem Perfect Recall werden die vorhergegangen Buckets mitgespeichert. Bei einem Imperfect Recall wird nur der aktuelle Bucket betrachtet und nicht die Vorhergegangenen. Der No-Limit-Bot im Poker Framework benutzt Imperfect Recall.

Durch diese Vereinfachungen ist es nicht mehr möglich das der CFR eine perfekte Strategie erstellt, jedoch kann der Algorithmus trotzdem gute Ergebnisse zu erzielen. Die Strategie wird nun mit vielen Iterationen des CFR berechnet, im Fall des hier beschriebenen Bots waren es 361 Milliarden Durchläufe. Die erstellte Strategie wird gespeichert. Ein Bot kann nun auf diese Strategie Tabelle zugreifen, die Wahrscheinlichkeiten auswerten und Unterberücksichtigung dieser eine Aktion wählen, die der Bot spielen soll. Der Bot reagiert bei diesem Verfahren jedoch nicht auf Strategien seiner Gegner, sondern spielt ausschließlich seine Strategie. Diese sollte zwar bei genügend Spielen immer das beste Ergebnis erzielen, berücksichtigt aber keine Fehler oder schlechte Strategien des Gegners um so kurzfristiger einen höheren Gewinn zu erzielen.

3.3 Counterfactual Regret Minimization

Beim Counterfactual Regret Minimization Algorithmus, kurz CFR, handelt es sich um einen Algorithmus, der ein Nash Equilibrium in einem Nullsummenspiel von zwei-Spielern approximiert. Ein Nash Equilibrium beschreibt eine Allgemeingültige Strategie, die über lange Zeit immer das beste Ergebnis erzielt. Während ein Nullsummenspiel ein Spiel bezeichnet in dem Verluste und Gewinne aufaddiert immer null ergeben. Beim Poker ist dass der Fall, da alle Gewinne, die ein Spieler bekommt, ein anderer Spieler verloren haben muss.

Der Algorithmus wird mit zwei Agenten mit der derselben Strategie initialisiert. Nun spielen die beiden Agenten gegeneinander. Zusätzlich wird der Nutzen, im Fall von Poker Gewinn bzw. Verlust von Chips, jeder Aktion berechnet. Dies ist möglich, da wir die Strategie unseres Gegners kennen. Nun wird für jeden Knoten ein Regret berechnet, der sich aus der Differenz der besten Aktion und der gewählten Aktion ergibt. Abhängig von diesem Ergebnis werden die Strategien im Spiel angepasst und der Agent spielt mit der neuen Strategie erneut gegen sich selbst.

Dieses Verfahren wird wiederholt, bis kaum noch Veränderungen in der Strategie vorhanden sind, kann aber auch nach einer bestimmten Anzahl an Iterationen abgebrochen werden. Bei genug Iterationen nähert sich die Strategie an ein Nash Equilibrium an. Genauer kann dies in [15] nachgelesen werden.

Es haben sich einige weitere Varianten des CFR Algorithmen entwickelt, wie z.B. Monte Carlo CFR [8] und Pure CFR [11] bei denen versucht wird, den Algorithmus schneller zur Konvergenz zu bringen und so weniger Iterationen zu brauchen. Einen CFR mit drei Spielern zu benutzen, ist zwar möglich und erzielt auch akzeptable Ergebnisse, jedoch verliert er die Grundlage, dass sich in jeden fall an ein Nash Equilibrium annähert [12].

3.4 Andere Arbeiten

Einige Arbeiten sind im Kontext diese Arbeit erwähnenswert. So ist Cepheus [14] der erste Bot gewesen der es geschafft hat Limit-Texas-Hold'em zu lösen und somit ein Nash Equilibrium für das Spiel zu erstellen.

Im Bereich No-Limit-Heads-Up-Texas-Hold'em schaffte es DeepStack [10] gute Ergebnisse gegen Profispieler zu erzielen. Dieser kombiniert die Ansätze eines CFR mit einem Neuronalem Netzwerk.

Eine weitere Interessante Arbeit ist [5], hier wurde versucht ein Drei-Spieler-Spiele auf ein Zwei-Spieler-Spiele abzubilden in einem Limit-Texas-Hold'em Spiel. Die Ergebnisse waren jedoch nicht sonderlich überzeugend, auch sind die angewendeten Techniken nicht auf No-Limit-Texas-Hold'em anwendbar.

Einige weitere Bot Varianten sind auf der ACPC Webseite [1] zu finden. Die meisten jedoch für Heads-Up-Limit und No-Limit-Texas Hold'em oder Drei-Spieler-Limit-Texas-Hold'em.

4 Mapping der Gamestates

Um einem Zwei-Spieler Bot einen gemappten Zustand eines Drei-Spieler-Spieles zu übergeben, müssen nun die Unterschiede zwischen den beiden Pokerarten überbrückt werden. Dafür müssen vor allem zwei verschiedene Anpassungen getroffen werden. Das Wegstreichen der hinzugefügten Position und die damit erforderliche Anpassung der Spieler und die Umwandlung der Aktionshistorie in einen sinnvollen und wenn möglich ähnlichen Ablauf. Ein beispielhafter Ablauf des Bots sieht wie folgt aus:

```
if hero is next then
    adjust gamestate for players;
    adjust gamestate for preflop;
    adjust gamestate for turn;
    adjust gamestate for turn;
    adjust gamestate for river;
end
Call Bot.getAction with new gamestate;
```

Dabei ist der grobe Ablauf unabhängig von der Street. In den folgenden Kapiteln wird beschrieben, wo die Probleme in den Anpassungen liegen und wie diese gelöst wurden.

4.1 Anpassung der Spieler

In einem Zwei-Spieler-No-Limit-Texas-Hold-em Poker Spiel gibt es genau zwei Positionen, Small Blind(SB) und Big Blind (BB), die in jedem Spiel von genau einer Person eingenommen wird. Dabei agiert, unabhängig von der Street, der SB immer zuerst. In einem Drei-Spieler-Spiel existieren nun zusätzlich zum Small- und Big Blind noch eine weitere Position, der Button (BU). Die Schwierigkeit besteht jetzt darin. Die zwei Gegner auf einen Spieler zu kürzen, und dabei die Reihenfolge der Aktion beizubehalten, so müssen die Positionen Small Blind, Big Blind und Button auf die Positionen Small Blind und Big Blind gekürzt werden.

Problematisch hierbei sind die unterschiedlichen Betting Reihenfolgen in einer Drei-Spieler-Situation zwischen der PreFlop Street und den nachfolgenden Streets. So ist PreFlop der BU der erste Akteur, aber post-PreFlop der letzte Akteur. In einer Heads-Up-PreFlop Situation übernimmt der Button die Position des SB, Post-PreFlop jedoch die Position des BB. Wenn ich nun die Positionen einfach beibehalte, erzeuge ich Situationen, in denen der Bot davon ausgeht, dass bereits eine Aktion getätigt wurde, dieses jedoch nicht der Fall ist, was zu einem zufälligen Verhalten des Bots führen kann. Um dieses Problem zu umgehen, müssen die Positionen der Spieler PreFlop und Post-PreFlop angepasst

Zum Anpassen der Position muss erst die Position des Heros ermittelt werden. Der Hero beschreibt den zu betrachten Spieler. Sitzt der Hero beim preFlop z. B. in der Position SB, muss bereits eine Aktion eines anderen Spielers getätigt worden sein. Wodurch der Hero als BB auftreten muss und der Mitspieler als SB auftreten. Für den Fall, dass ein Spieler in der Street zuvor gefoldet hat, müssen unter Umständen weitere Positionsanpassungen getroffen werden. In der Tabelle 3 wird diese Positionsanpassungslogik dargestellt. Die erste Spalte zeigt die Postion des Heros in einer Drei-Spieler-Situation. Gegner Fold Position zeigt die Position an, in der ein Gegner gefolded hat. Wenn das Feld leer ist, sind noch alle Spieler Aktiv oder haben erst in der aktuellen Street gefolded. In der Zeile “Mapping” wird beschrieben, welche Rolle der Hero und der Opponent in dem gemappten Spielzustand annehmen

Tabelle 3: Positionsanpassungslogik

3-Spieler Situation			Mapping	
Hero Position	Gegner Fold Position	Street	Hero Position	Opponet Position
SB	-	Preflop	BB	SB
BB	-	Preflop	BB	SB
BU	-	Preflop	SB	BB
SB	-	PostPreFlop	SB	BB
BB	-	PostPreFlop	BB	SB
BU	-	PostPreFlop	BB	SB
SB	BU	PostPreFlop	SB	BB
BB	BU	PostPreFlop	BB	SB
SB	BB	PostPreFlop	SB	BB
BU	BB	PostPreFlop	BB	SB
BB	SB	PostPreFlop	SB	BB
BU	SB	PostPreFlop	BB	SB

Ein Sonderfall tritt ein, wenn die erste Aktion eines Gegners im Flop ein Fold ist, da nun eine perfekte Heads-Up-Situation erstellt wurde, da die zu kürzende Position BU, ohne Einsatz getätigt zu haben, gefolded hat. So kann schon in der ersten Street die Position BU gestrichen werden.

Durch das Verändern der Position eines Spielers zwischen den Streets ist es nicht mehr möglich die PreFlop Aktionen dem richtigen Spieler zuzuweisen. Dies muss bei der Auswahl des Zwei-Spieler-Bots, der den gemappten Zustand auswerten soll, beachtet werden, damit der Bot hier durch das Mapping keine falschen Annahmen trifft. Im Fall des in 2.2 beschriebenen CFR Bots ist dies durch den imperfekten Recall gegeben.

4.2 Anpassung der Aktionshistories

Bei der Anpassung des Betting-Strings ist die größte Schwierigkeit die Umwandlung in einen ähnlichen Zwei-Spieler-String. So sind Zwei-Spieler Strings meistens eine Aneinanderreihung von Raises, gefolgt von einem Call oder Fold, bei Mehrspieler Stings können alle Aktionen auftreten, ohne dass die Street beendet sein muss.

Ein weiterer sinnvoller Ansatz ist das Beibehalten der eigenen Aktionen, da diese von dem Bot ausgewählt wurden und es so sinnvoll ist auch genau diese Aktion weiterhin zu verwenden. Wenn möglich sollte diese im weiteren Verlauf berücksichtigt werden.

Die Betting-Strings müssen so lange angepasst werden, bis in der Street zuvor ein Spieler Gefolded hat. Sobald dies jedoch geschehen ist, können alle weiteren Aktionen 1 zu 1 übernommen werden, da jetzt die Anzahl der Spieler mit einem Zwei-Spieler-Spiel übereinstimmt und daher keine falschen Zustände entstehen können. Die Streets, in der besagter Spieler gefolded hat und die Street zuvor müssen, jedoch weiterhin gemappt werden. Folded ein Spieler Beispeilweise auf dem Turn, müssen die Aktionshistories des Preflops, des Plops und des Turns weiterhin gemappt werden. Die Aktionshistorie des Rivers jedoch kann ohne Mapping übernommen werden.

4.2.1 Look-Ahead Tabel

In Normalfall kommen auf eine Aktion des Heros zwei Aktionen von Gegenspielern. So müssen jeweils zwei Aktionen auf eine Aktion abgebildet werden. Der Betting String “R300 C R500” wäre in einer Zwei-Spieler-Situation so nicht darstellbar, da nach dem ersten Call die Street beendet wäre. Zu vereinfachung werden Raise, ungeachtet der Höhe, immer als ‘R’ angezeigt. Calls werden weiterhin als ‘C’ und Folds als ‘F’ dargestellt. Vereinfacht sieht die Aktion Historie von oben nun so aus “RCR”.

Geht man davon aus, dass der Hero an erster Position den Raise getätigt hat und die Aktion des Heros beibehalten werden soll, ist der Rasie “R300” gesetzt. So muss also nur noch “CR” auf einen Spieler abgebildet werden. Die einzige logische Abbildung ist ein Raise, da in einer Heads-Up-Situation die Street nach einem Call beendet wäre, sie in einem

Drei-Spieler-Spiel jedoch noch nicht beendet ist. Dadurch ergibt sich die in Tabelle 4 dargestellte Look-Ahead Tabelle. Diese beschreibt das Pattern von zwei Aktionen der Gegner und gib dazu ein passendes Mapping zurück.

Tabelle 4: Vorläufige Look-Ahead Tabelle

Aktionen	Mapping	Besonders/ Beispiel
CC	C	beendet die Street, außer am Beginn einer Street
CR	R	c r300 $\rightarrow r300$
CF	C	beendet die Street außer am Beginn einer Street
RC	R	r 300c $\rightarrow r300$
RR	R	r300 r500 $\rightarrow r500$
RF	R	r300 f $\rightarrow r300$
FC	C	f c$\rightarrow c$
FR	R	f r300 $\rightarrow r300$

“FF” muss nicht betrachtet werden, da nachdem zwei Spieler gefolded haben das Spiel vorbei ist und der Bot nicht mehr nach einer Aktion gefragt wird. Für den “RR” fall wird immer der zweite Raise als Wert ausgewählt, da dies der Einsatz ist, der mindestens aufgebracht werden muss um weiter an dem Spiel teilzunehmen. Beispiel: “RxRy”, wobei x $<$ y, wird als Raise Amount y ausgewählt.

Sobald einer der Gegenspieler folded können die Aktionen einfach weiter übernommen werden, da jetzt nur noch zwei Spieler Aktiv am Spiel teilnehmen und die weiteren Aktionen dieselbe Form wie in ein Zwei-Spieler-Spiel besitzen. Das macht die “FC” und “FR” Darstellungen im Look-Ahead überflüssig.

Für den Fall, dass der Hero als Zweites agiert, treten auch andere Muster auf, da eine alleinstehende Aktion am Anfang des Strings, die nicht vom Hero getätigt wurde, ausgewertet werden muss. So muss auch “C”,”R” und “F” ausgewertet werden, dies ist jedoch kein Problem, da die Aktionen Call und Raise einfach so übernommen werden können und ein Fold einfach gestrichen werden kann. Dieses Verhalten sorgt für einen Spezialfall des Bots. Für den Fall, dass der Hero nicht in der Position des BU ist und der Gegner als erste Aktion im Preflop folded können alle weiteren Aktionen 1 zu 1 übernommen werden, ohne gemappt werden zu müssen. Da der BU weder einen Einsatz getätigt hat noch einen Blind bezahlen musste, sind auch die Potsizes zwischen dem gemappten- und dem Drei-Spieler-Zustand gleich. Da, wie in 4.1 beschrieben, in diesem Fall auch die Position des BU schon im PreFlop gestrichen werden kann, entsteht in diesem Fall ein perfekter Zwei-Spieler-Zusatnd, welche auch in den weiteren Streets nicht angepasst werden muss, bis auf den Fold der am Anfang der PreFlop-Historie gelöscht werden muss.

Beachtet werden muss, dass der Algorithmus auch eine leere Actionhistorie bekommen kann, diese kann einfach als Leer-String gemappt werden. Dadurch ergibt sich eine überarbeitet Look-Ahead Tabelle dargestellt in Tabelle 5.

Tabelle 5: Look-Ahead Tabelle

Aktionen	Mapping
C	C
R	R
F	
CC	C
CR	R
CF	C
RC	R
RR	R
RF	R

4.2.2 Mapping Algorithmus

Mit dem look-Ahead Tabel aus 4.2.1 ergibt sich nun folgender Ablauf zum Mappen der Actionshistorie:

- Schritt 1: Unterteilen für den Fall eines Foldes im String
  Da nach einem Fold eines Gegners der String 1 zu 1 übernommen werden kann, muss der Teil nach dem Fold nicht weiter betrachtet werden. So kann er einfach vom Rest des Strings abgespaltet werden.

- Schritt 2: Position des Heros ermitteln

  Da die Aktion des Heros wenn möglich übernommen werden soll, muss die Position des Heros im String ermittelt werden.

- Schritt 3: Aktionen in Hero-Aktionen und in Non-Hero-Aktionen unterteilen

  Den Reststring in Hero- und Non-Hero-Aktionen unterteilen, wobei die Teile die in Hero-Aktionen unterteilt wurden, immer die Länge 1 haben müssen und die Non-Hero-Aktionen die Länge 1 oder 2 haben.

- Schritt 4: Über Look-Ahead Tabelle Gegneraktion bestimmen
  Die Non-Hero Aktionen des gemappten Zustandes über Look-Ahead Tabelle mappen.

- Schritt 5: Alle Teile Zusammensetzen.

  Alle Teilstrings werden wieder zusammengesetzt und es sollte eine gültige Zwei-Spieler String entstanden sein.

Beispiele 1: C R200 C R300 R600 R1500 R3000 C C

Schritt 1)
Da es in dem String keinen Fold gibt, geschieht nichts. Für den Fall, dass es einen Fold im String gibt, wird dieser Teil abgespaltet und muss nicht verändert werden.

C R200 C R300 R600 C R1500 R3000 C C

Schritt 2)
Rein an dem String ist die Position nicht zu bestimmen, für das Beispiel treffen wir die Annahme, dass der Hero in Position 1 sitzt. Die eingefärbten Aktionen zeigen die Aktion des Heros.

C	R200 C	R300	R600 C R1500	R3000	C C

Schritt 3)
Der String wird in Non-Hero- und Hero Aktionen eingeteilt. Die eingefärbten Teile sind weiterhin die Hero Aktionen.

C	R200 C	R300	R600 R1500	R3000	C C

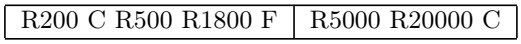
Schritt 4)
Anpassungen der Non-Hero Aktionen über den Look-Ahead Tabelle, so hat “R200 C” die Form “RC” und wird mit blick auf Tabelle 5 zu einem R ausgewertet, die Höhe beträgt 200, da dies der einzige Raise in der Aktionsfolge ist. “R600 R1500” hat die Form RR ergibt also ebenfalls einen Raise. Als höhe wird immer der zweite Wert gewählt in dem Fall 1500.

C	R200	R300	R1500	R3000	C

Schritt 5)
Nun werden die einzelnen Schritte wieder zusammengefasst und es ergibt sich der gemappte Zwei-Spieler-Aktionshistories.

C R200 R300 R1500 R3000 C

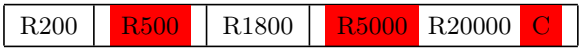
Beispiel 2: R200 C R500 R1800 F R5000 R20000 C
Schritt 1)



Schritt 2)
Hero in Position 3

Schritt 3)

Schritt 4)



Schritt 5)

4.2.3 Anpassung der Pot Size

Da ein R400 nicht bedeutet Raise auf 400, sondern Raise auf 400 von seinem Stack, sorgt der Algorithmus dafür, dass der CFR mit einem gemappten Spielsituation Konfronterit in der die Potsize nicht korrekt ist.

Beispiel 3:
R200 R300 R600 R2000 C R8000
wird mithilfe des Mappings auf
R200 R600 R2000 R8000
abgebildet. Wobei die rot markierten Aktionen die Hero-Aktionen anzeigen.

Tabelle 6: Drei-Spieler Spiel Potsize

	Einsatz			
Runde	Hero	Opponent1	Opponent2	Potsize
Blinds		50	100	150
1	200	300	600	1100
2	2000	2000	8000	12000

Tabelle 7: Mapping Potsize

	Einsatz		
Runde	Hero	Opponent	Potsize
Blinds	50	100	150
1	200	600	800
2	2000	8000	10000

Die reale Potsize beträgt 12000 Chips. Die Potsize des Mappings beträgt lediglich 10000 Chips. Um die Potsize anzupassen, müssen die Raise Amounts angeglichen werden. Am einfachsten funktioniert dies indem die Differenz zwischen der realen Potsize von der Potsize des Mappings abgezogen wird und die Differenz auf den letzten Raise

aufgerechnet wird. Für den Fall, dass die letzte Aktion ein Call ist, wird die Differenz halbiert und auf den letzten Raise aufaddiert. Eine Ausnahme ist hier, wenn der Raiseamount + Differenz größer gleich der Stacksize eines Spielers ist. In diesem Fall wird die Potsize nicht angepasst, da der Spieler sonst mehr Chips setzen müsste als er hätte. Für den Fall, dass bisher kein Raise getätigt wurde, kann es sein das auch hier die Potsize sich unterscheided, dann wird sie logischerweise ebenfals nicht angeglichen.

Für das Beispiel 3 ergibt sich also eine Differenz zwischen realer Potsize und Potsize des Mappings von 2000. Dieser Wert muss auf den letzten Raise aufaddiert werden, in diesem Fall R8000. Der Potsize angepasste String sieht wie folgt aus “R200 R600 R2000 R10000”.

Durch diese Anpassung treten nun jedoch einige Verschiebungen auf. So wurde zwar die Potsize angepasst, jedoch wurde der Amount-to-Call erhöht. Der Amount-to-Call beschreibt den Unterschied zwischen meinem bisherigen Einsatz und dem höchsten Einsatz, der bereits getätigt wurde, dieser Wert ist von Bedeutung, da er den minimalen Betrag beschreibt, der getätigt werden muss, um noch weiter an der aktuellen Hand teilzunehmen. Die Erhöhung des Wertes sorgt dafür, dass der Bot davon ausgeht einen höheren Einsatz callen zu müssen, als er eigentlich müsste, um weiter am Spiel teilzunehmen. Dieses kann zu einer Veränderung der Strategie führen und dadurch zu einem veränderten Ergebnis. Durch die internen Mappings des CFR Bots kann es jedoch sein, dass dadurch keine großen Unterschiede entstehen, da die Aktionen auf die selbe Aktion abgebildet wurden.

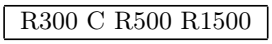
Die Änderung an der Aktion des Heros ist zwar möglich aber nicht relevant, da die Hero Aktion nur dann verändert wird, wenn die Street bereits beendet ist und der Bot, wie in 4.1 beschrieben, nicht die genau Abfolge in den beendeten Streets betrachten sollte.

4.2.4 Hero Call Problem

Probleme bereitet der Bot für den Fall eines Calls des Heros. Sollte der Hero gecalled haben und einer der andern Bots in derselben Street noch mal geraised haben, muss der Bot ein weiteres Mal agieren. Dadurch entstehen unmöglicher Spilzustand die nicht interpretiert werden können. Wenn die erste Aktion des Bots in einer Street ein Call ist, gibt es jedoch keine Probleme, da die Street in einem Heads-Up-Spiel noch nicht beendet ist.

Beispiel 4: R300 C R500 R1500 mit Hero in Position 2.

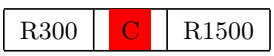
Schritt 1)



Schritt 2)
Hero in Position 2

Schritt 3)

Schritt 4)



Schritt 5)

Der erzeugte Spielablauf kann in einem Heads-Up-Texas-Holdem Spiel so nicht vorkommen, da die Street nach dem Call des Heros beendet sein müsste. Zum Beheben dieses Problems muss also der Call des Heros entfernt werden und

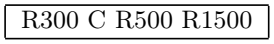
durch eine andere Aktion ersetzt werden. Was dazu führt, dass es nicht mehr möglich ist, die Aktion des CFR Bots zu übernehmen. Einige Möglichkeiten mit dem Problem zu verfahren wären nun:

- Variante 0: Wegstreichen des Calls

  Hierbei wird der Call einfach gestrichen. Damit dies funktioniert, muss jedoch auch die vorherige Aktion des Gegners gestrichen werden, da sich sonst die Reihenfolge der Spieler verändern würde.

  Beispiel 4 mit Variante 0:

  Schritt 1)

  Schritt 2)
  Hero in Position 2

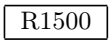
  Schritt 3)

  Schritt 4)

  Schritt 5)

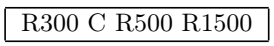
  Diese Herangehensweise sorgt jedoch dafür, dass sich der Amount-to-Call zwischen dem realen Spiel und dem Mapping stark unterscheiden kann. Wie im Beispiel zu sehen geht der Bot im Mapping von einem Einsatz von 100 (Big Blind) aus und muss mindestens einen Einsatz von 1400 leisten. In dem realen Spiel hat der Bot bereits einen Einsatz von 300 getätigt und muss nur mindestens einen Einsatz von 1200 tätigen.

- Variante 1: Call in MinRaise umwandeln:

  Hierbei wird der Call des Spielers in den kleinstmöglichen Raise umgewandelt.

  Beispiel 4 mit Variante 1

  Schritt 1)

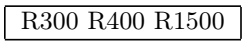
  

  Schritt 2)
  Hero in Position 2

Schritt 3)

Schritt 4)

Schritt 5)

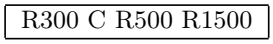


Vorteil im Gegensatz zu Variante 0 ist die Übereinstimmung der Anzahl der Aktionen des Spiels, jedoch wird auch hier die Amount-to-Call verändert. Im Gegensatz zu Variante 0 jedoch in die entgegengesetzte Richtung, da der Bot nun von einem höheren Einsatz ausgeht, als er getätigt hat.

- Variante 2: Raise des Gegners übernehmen I:

Hier wird versucht den Raise des Gegners zu übernehmen. Dies ist jedoch nur möglich, wenn beide nachfolgenden Aktionen Raises sind. Falls das nicht der Fall ist, wird Variante 0 ausgeführt und der Call gestrichen.

Beispiel 4 in Variante 2
Schritt 1)

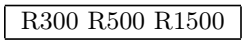


Schritt 2)
Hero in Position 2

Schritt 3)

Schritt 4)

Schritt 5)



Nachteil an dieser Variante ist, dass für den “RR” Fall der Amount-to-Call noch stärker abweicht als bei Variante 1. Im Beispiel geht der Bot nach dem Mapping davon aus, dass er schon 500 Chips investiert hat, in Wirklichkeit sind es jedoch nur 300. Dafür ist die Potsize des Mappings näher an der wirklichen Potsize.

- Variante 3: Raise des Gegners übernehmen II

  IV agiert bei einem “RR” Fall wie Variante 2. Für den Fall, dass dieser jedoch nicht eintritt, führt er Variante 1 aus.

Alle diese Varianten haben Nachteile sind aber nötig um einen funktionsfähigen Spielzustand zu erstellen.

4.3 Zusammenfassend

Während der Implementierungen entstanden mehrer Mappings in zwei Kategorien: Potsize Anpassung und Hero Call Aktion. Dargestellt in Tabelle 8 und 9. Die Mappings aus Kategorie I sind mit den Mappings aus Kategorie II kompatibel. Es wurden also 8 verschiedene Bots erzeugt. Teilweise mit größeren und kleineren Anpassungen. Die Bot-Varianten werden im folgenden Teilen immer mit den Kürzeln aus den Tabellen abgekürzt. So heißt beispielhaft der Bot, der die Potsize nicht anpasst und Calls streicht, N;0.

Tabelle 8: Potsize Anpassung

Kürzel	Bot	Verlust	Vorteil
A	Anpassen der Potsize	Erhöhung der Raises, Veränderung von Pot Odds und Amount-to-Call	Korrekte PotSize
N	Nicht anpassen der Potsize	Potsize ist geringer als in realen Spiel	Korrekte Raises, Amount-to-call Korrekt

Tabelle 9: Hero Call Aktionen

Kürzel	Bot	Verlust
0	Wegstreichen des Calls	Erhöhung des Amount-to-Call
1	Call in MinRaise umwandeln	leichte Verringerung des Amount-to-Call
2	Raise des Gegners übernehmen I	leichte bis starke Verringerung des Amount-to-Call bei “RR”, sonst Erhöhung des Amount-to-Call
3	Raise des Gegners übernehmen II	leichte bis starke Verringerung des Amount-to-Call bei “RR”, sonst leichte Verringerung des Amount-to-Call

Die umgewandelten Spielsituationen werden an den CFR Bot im TU Darmstadt Poker Framework weitergeleitet und dieser gibt sein Ergebnis zurück. Theoretisch ist es auch möglich jeden anderen Bot zu verwenden, der für Heads-Up-No-Limit-Texas-Hold’em Poker konstruiert wurde und nicht die gesamte Historie betrachtet.

Das Testen der gemappten Bot-Varianten erweist sich als schwierig, da keine starken Drei-Spieler-No-Limit-Texas-Hold'em Bots zu Verfügung stehen und daher nur primitivere Bots zum Auswerten benutzt werden können. Dennoch gibt es verschiedene Herangehensweisen, den Bot zu testen. So kann man den Bot gegen sehr primitive Bots wie z.B. MinRaise oder AllInBots testen. Diese spielen jedoch kein "sinnvolles" Pokern, was die Aussagekraft stark eingegrenzt . Nichtsdestotrotz sollten der gemappte CFR mit solchen primitiven Bots zurechtkommen und diese schlagen können. Ein deutlich besserer Bot ist da schon der EVBot. Für erfahrene Spieler ist dieser Bot jedoch auch kein großes Hindernis, da an den Aktionen des Bots die Handkarten analysiert werden können. Des Weiteren spielt der Bot sehr aggressiv. Dies liegt daran, dass der Bot seine Handstärke mit der All-in Equity bemisst, wodurch eine Hand mit einer 51% Gewinnwahrscheinlichkeit als stark bewertet wird. Dies können gute Spieler ausnutzen, indem sie ihre Strategie auf dieses Verhalten anpassen. So sollte ein spielstarker Bot auch den EVBot schlagen können.

Für die Bots können mehrere Informationen ausgewertet werden:

- Gewinn/Verlust:
  Alle Bots können bewertet werden, indem die Gewinne eines Bots nach einem Testlauf miteinander verglichen werden. Dies ist möglich mit dem reinen Gewinn oder dem genauen bb/100 Wert, welcher den Big Blind Gewinn in Hundert Spielen bezeichnet. Ein wünschenswertes Ergebnis ist ein positiver Gewinn, da er bedeutet, dass der Bot den anderen Bot schlägt und somit über lange Sicht Profit erzielt. Der Gewinn muss hoch genug sein, um eine statistische Signifikanz beizubehalten. Für einfache Bots wie einem All-In Bot sollten die Gewinne deutlich höher ausfallen. Wie hoch der Gewinn sein sollte ist schwer zu beantworten. So gewann als Beispiel DeepStack mit 49 bb/100 im Schnitt gegen verschiedene Profispieler [2] [10].

- Gewinn/Verlust Verteilung:
  Bei diesem Test werden nicht die reinen Gewinne herangezogen, sondern geschaut wie hoch die einzelnen Verluste und Gewinne sind und wie oft diese auftreten. Auch ergeben sich hier Informationen wie Anzahl der gewonnenen bzw. verlorenen Hände, oder der Prozentsatz der gewonnen All-ins. Hierbei sollen Unterschiede zwischen den verschiedenen Bots deutlich gemacht werden.

- Aktion Häufigkeiten:
  Bei diesem Test wird geschaut, wie oft welcher Bot welche Aktion tätigt. Dieses kann zusätzlich noch nach Street gefiltert werden. Hier sollten Unterschiede zwischen den verschiedenen Bots deutlich gemacht werden.

- Aktionsvariation:
  Bots die auf dem CFR basieren können miteinander verglichen werden indem die von CFR berechneten Aktionswahrscheinlichkeiten auf Übereinstimmung getestet werden. Hierbei wird zu dem Test noch ein weiterer Bot einer anderen Variante gestartet. Bei jeder Entscheidung werden nun beide Bots befragt. Da nicht immer das Ergebnis mit der höchsten vom CFR errechneten Wahrscheinlichkeit gewählt wird [11], kann anhand der höchsten Wahrscheinlichkeit der zwei verschiedenen CFR Varianten ein Unterschied bestimmt werden.

Zum Testen eines Bots müssen jeweils zwei gegnerische Bots gewählt werden. Hier gibt es die Möglichkeit den zu testenden Bot gegen zwei gleiche Bots zu testen. So können die Ergebnisse mit Vorsicht auch mit dem Heads-Up CFR Bot verglichen werden. Eine weitere Möglichkeit wäre das Spielen gegen zwei verschiedene Bots. Jeder der 8 verschiedenen Bot Varianten aus 4.3 werden getestet.

Mögliche Tests wären:

- Bot vs AllinBot vs AllinBot
- Bot vs EVBot vs EVBot
- Bot vs EVBot vs MinRaiseBot
- Bot vs EVBot vs AllinBot
- Bot VarianteX vs Bot Variante Y vs Bot Variante Y

Alle Tests werden mit tausend verschiedenen Händen in Duplicate-Modus, bei welchem die Positionen und die Karten in jeder möglichen Konstellation vergeben werden, durchlaufen. So werden pro Testdurchlauf 6000 Händen ausgewertet. Um die Tests bessere mitteinander Vergleichen zu können, wird ein festgelegter Seed ausgewählt. Ein Seed beeinflusst die Zufallsparameter des Spiels. So werden bei einem Spiel immer dieselben Karten ausgeteilt, was zu besser Vergleichbarkeit zwischen verschiedenen Bots führt. Hierbei sollte jedoch darauf geachtet werden, dass kein zu großes Overfitting geschieht. Dies wird zumindest durch den Duplicate-Modus stark verringert, da so ein Spieler nicht durch Glück deutlich bessere Karten als seine Gegner ausgeteilt bekommen kann. Der Seed beeinflusst jedoch nicht die Zufallsparameter des CFR Bots, wodurch dieser nicht immer dieselbe Aktion auswählt, was bei zwei gleichen Spielzuständen zu anderen Ergebnissen führen kann.

Das Ziel der Tests ist es zu schauen, wie stark die gemappten Varianten des CFRs sind. Wie gut sie sich gegen die verschiedenen Bots behaupten können. Zusätzlich sollte analysiert werden wie und warum sie sich gut oder schlecht schlagen. Ebenfalls können die verschiedenen Varianten miteinander verglichen werden.

6 Ergebnisse

6.1 Gewinn/Verlust

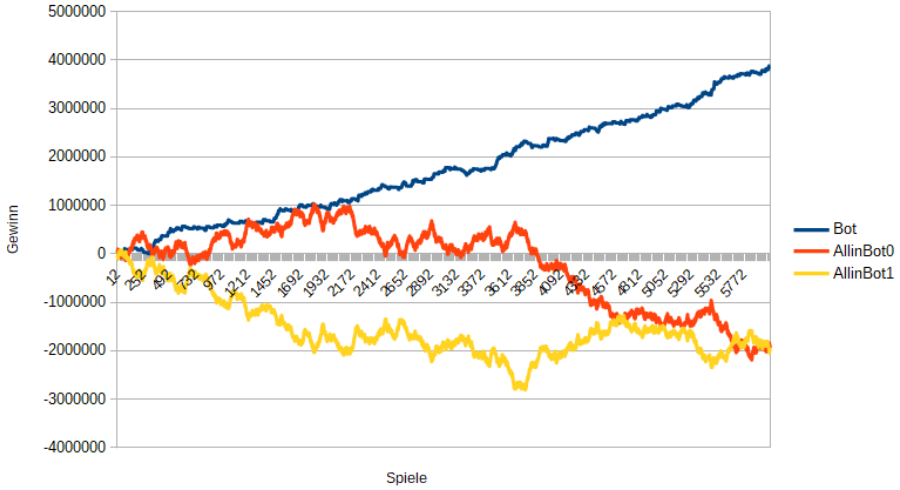


Abbildung 1: Grafische Darstellung von Gewinne/Verluste A;0 vs AllinBot vs AllinBot

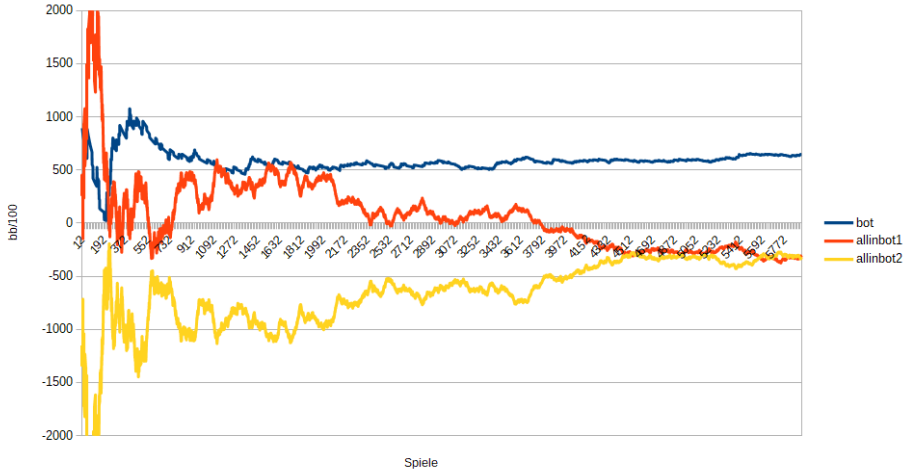


Abbildung 2: Grafische Darstellung von Gewinne/Verluste A;0 vs AllinBot vs AllinBot in bb/100

In Abblidung 1 ist der Gewinn des A;0 Bots gegen zwei AllinBots über 6000 Spiele verteilt zu sehen. Der Gewinn der Bots beträgt 3882700 Chips. Während beide AllinBots einen Verlust von 1940000 Chips erzielen. Abblidung 2 zeigt den Big Blind Gewinn pro 100 Spiele. Dieser beträgt 647 Big Blinds pro 100 Spiele, während die beiden Bots einen Verlust von 323 bb/100 haben.

Im Vergleich dazu zeigt Abblidung 3 den Gewinn des CFR gegen einen All in Bot, hier jedoch nur über 2000 Spiele, da der Duplicate Modus bei nur zwei Spielern die Hände nur in zwei Positionen gespielt werden können. Der Gewinn des CFR Bots beträgt 384 bb/100.

Es gibt zwischen den gemappten Bot Varianten nur sehr geringe Unterschiede, die durch die nicht immer gleiche Aktionsauswahl des CFR erklärt werden können. Die Ergebnisse sind so ähnlich, da nur vier verschiedene Abläufe gemappt werden müssen:

- “ ”

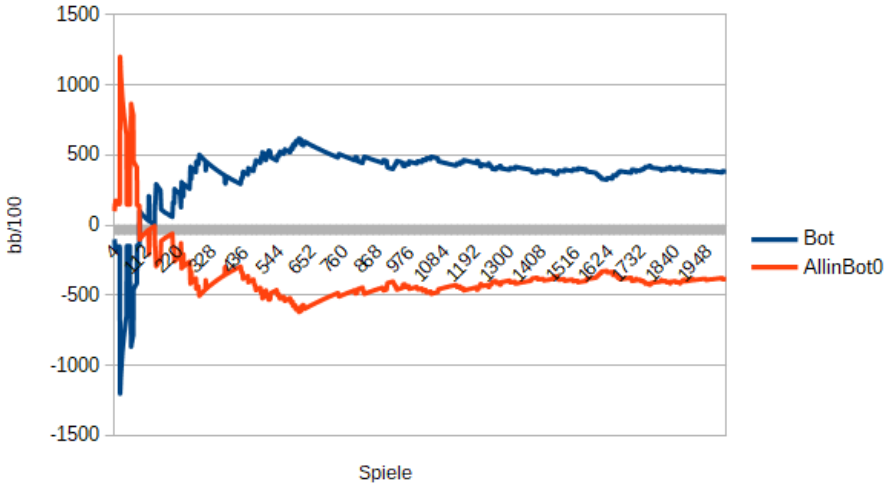


Abbildung 3: Grafische Darstellung von Gewinne/Verluste CFR vs AllinBot in bb/100

- R20000
- R20000 C
- (Spieleraktion) R20000 C

Diese Aktionen werden von allen Bot Varianten gleich gemappt.

Wenn Abblidung 3 mit Abblidung 2 verglichen wird ist zu erkennen, dass der Verlust eines AllinBots in Abbildung 2 sehr ähnlich zu dem Verlust des AllinBots aus Abbildung 3 ist, was dafür spricht, dass die Bots trotz des veränderten Spielzustandes noch immer ähnliche Entscheidungen treffen.

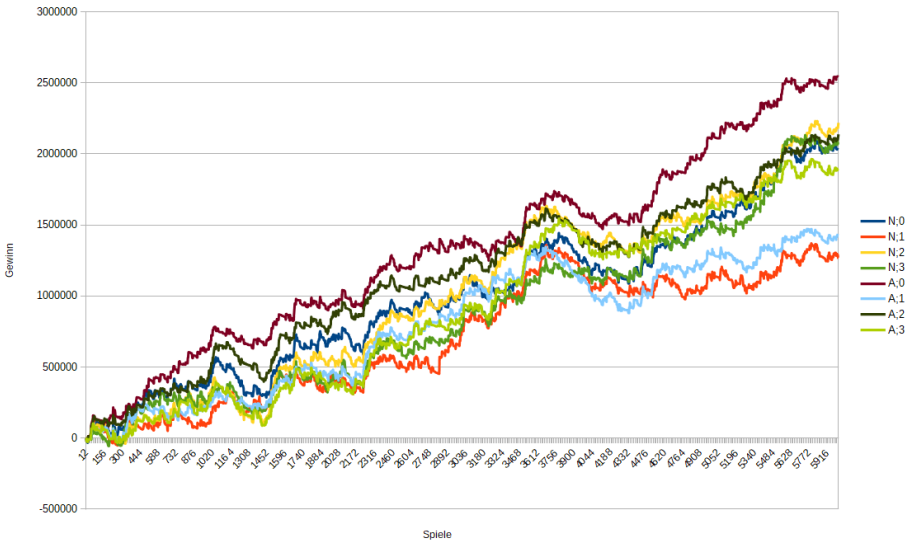


Abbildung 4: Grafische Darstellung von Gewinnen/ Verlusten aller gemappte Bot-Varianten gegen zwei EVBots

Abblidung 4 und 5 zeigen die Gewinne aller gemappten CFR Bot-Varianten gegen jeweils zwei EVBots. Die Kürzel wurden aus 4.3 entnommen. Zu erkennen sind drei verschiedene Zonen von Ergebnissen: Die beiden Bot Varianten mit dem kleinsten Gewinn sind die Bots A;1 und N;1(Calls werden zum minraise umgewandelt). Die beiden haben einen Gewinn von ca. 1300000 Chips bzw. einen Gewinn von ca. 220 bb/100. Die zweite Gruppe besteht aus den Bots

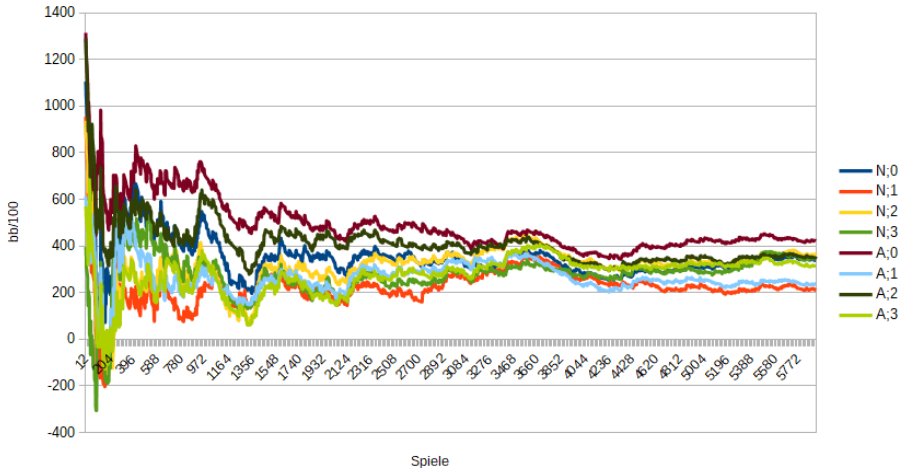


Abbildung 5: Grafische Darstellung von Gewinnen/ Verlusten aller gemappte Bot-Varianten gegen zwei EVBots in bb/100

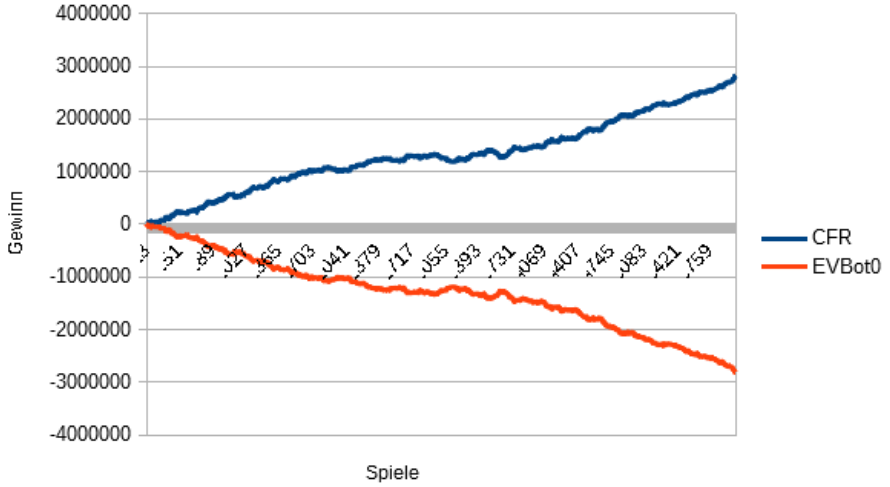


Abbildung 6: Grafische Darstellung von Gewinnen/ Verlusten CFR gegen EVBot

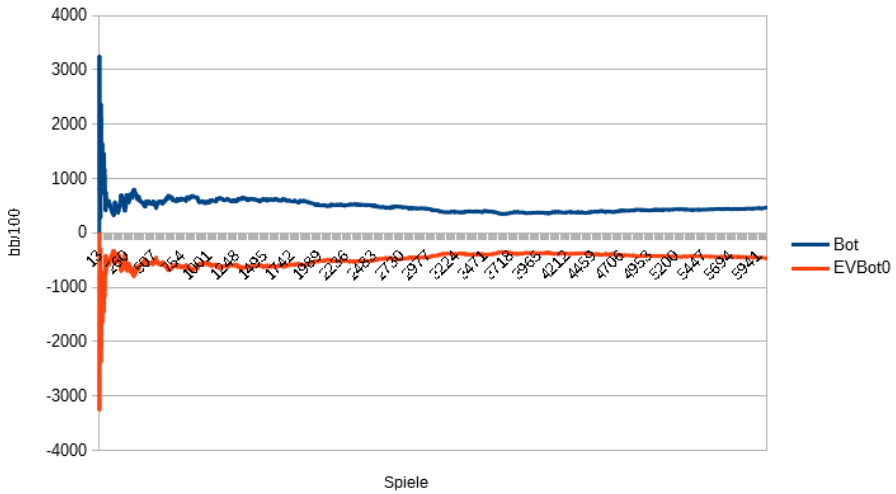


Abbildung 7: Grafische Darstellung von Gewinnen/ Verlusten des CFR gegen EVBots in bb/100

N;0, N;2, N;3, A;2, A;3 der Gewinn liegt zwischen 1900000 und 2100000 Chips bzw. 313 und 369 bb/100. Der Bot mit dem höchsten Gewinn ist A;0 (Anpassen der Potsize und wegstreichen der Call), dieser hat einen Gewinn von 2500000 bzw. von 423 bb/100.

Auffällig hierbei ist, dass Call-Modus 1 (Calls werden zu minraises umgewandelt) am schlechtesten abschneidet. Eine weiter Auffälligkeit ist das A;0 besser abschneidet als N;0. Des Weiteren kann man erkennen, dass das Ergebnis der andern Bots sehr nahe beieinander liegt. Im Vergleich dazu sind die Gewinne des CFR Bots mit 2814000 Chips oder 469 bb/100 ein wenig höher als die Ergebnisse des A;0. Die Verluste eines einzigen EVBots verdoppeln sich jedoch im Vergleich zu dem A;0.

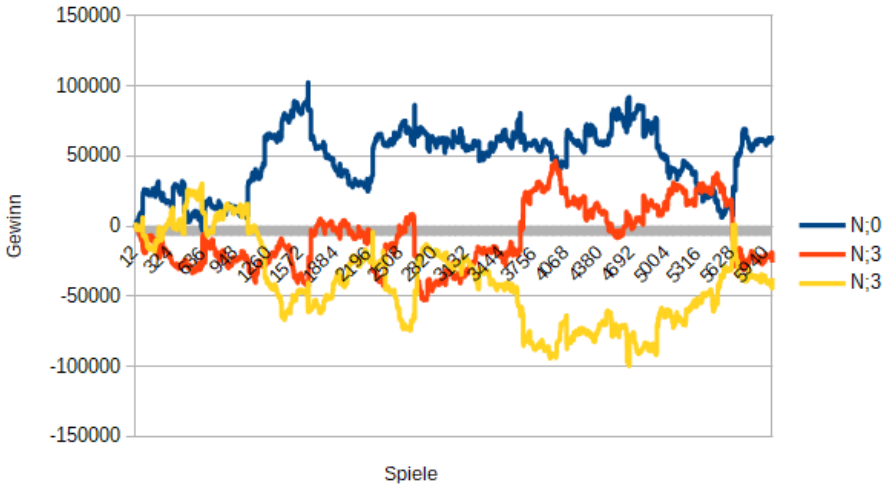


Abbildung 8: Beispielhafte Darstellung von Gewinnen/Verlusten N;0 vs N;3 vs N;3 aus Tabelle 10

Tabelle 10: bb/100 N;0 gegen 2 Bots der anderen Varianten

	bb/100 N;0	bb/100 Bot 1	bb/100 Bot 2
N;1	3,44	3,50	-6,94
N;2	-8,54	14,23	-5,70
N;3	2,66	7,84	-10.50
A;0	-1,32	-5,29	6,61
A;1	8,33	-23,57	15,24
A;2	8,22	-2.06	-6.16
A;3	10.49	-4,31	-6,175

Beim Vergleich der Bot Varianten untereinander treten einige Probleme beim Auswerten auf. So sind die Ergebnisse der Bots untereinander schwer miteinander zu vergleichen. Wie man in Tabelle 10 erkennen kann, sind die Ergebnisse im Vergleich zu den Spielen gegen die anderen Bots deutlich geringer. Sie sind nur selten über 10 bb/100. Problematisch sind jedoch die Ergebnisse der jeweils selben Bots. So verliert der Erste A;1 Bot zwar 23 bb/100 der Zweite A;1 gewinnt jedoch 15 bb/100, obwohl beide Bots gleich eingestellt wurden. Dies lässt darauf schließen, dass die Ergebnisse für die Anzahl der Spieldurchläufe nicht statistisch signifikant sind.

Ein weiteres interessantes Verhalten des Bots ist zu erkennen, bei den Ergebnissen zwischen dem EVBot, dem AllinBot und einer gemappten CFR Variante. Der EVBot hat einen Gewinn von 4400000 oder 733 bb/1000. Der AllinBot einen Verlust von 7400000 oder 1237 bb/100. Der A;0 hat einen Gewinn von 2900000 oder 496 BB. Dargestellt in Abbildung 9. Hier ist das Verhalten des CFR zu erkennen. Dieser gewinnt zwar gegen den AllinBot, hat aber einen geringeren Gewinn als der EVBot. Dies liegt daran, dass die Strategie des CFR nicht an den Gegner anpasst.

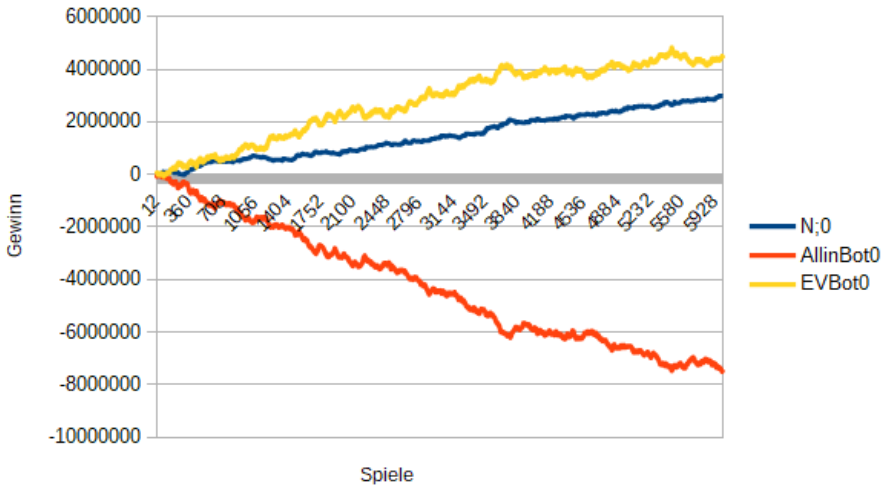


Abbildung 9: Grafische Darstellung von Gewinnen/Verlusten von EVBot vs AllinBot vs N;0

Alleine am Gewinn können jedoch die Ergebnisse nicht näher erklärt werden, da sie nicht aufzeigen, wie die Gewinne zustande kommen. Hierfür wurden noch weitere Tests durchgeführt.

6.2 Gewinn/Verlust Verteilung

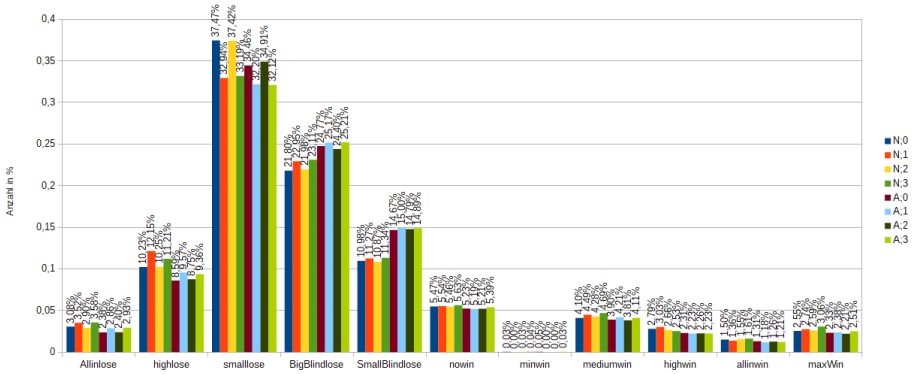


Abbildung 10: Grafische Darstellung der Gewinn-/Verlustverteilung des gemappte CFR-Varianten gegen zwei EVBots in Kategorien: Prozentuale Anzahl

In Abbildung 10 sind die Gewinne und Verluste der Bots nach Höhe summiert. Auch hier waren die Gegner jeweils zwei EVBots. Die Kategorien beschreiben folgende Situation:

- SmallBlindLose: Verlust des Small Blind in Höhe von 50 Chips.
- BigBlindLose: Verlust des Big Blinds in Höhe von 100 Chips.
- SmallLose: Verlust zwischen 101 und 1500 Chips, was meistens einem Raise oder einem Call und darauf einem Fold entspricht.
- HighLose: Fold nach mehreren Raises eines Spielers..
- AllInLose: Lose nach einem All in..
- NoWin: Fold in der Position des Buttons, was weder einem Win noch einem Lose entspricht.

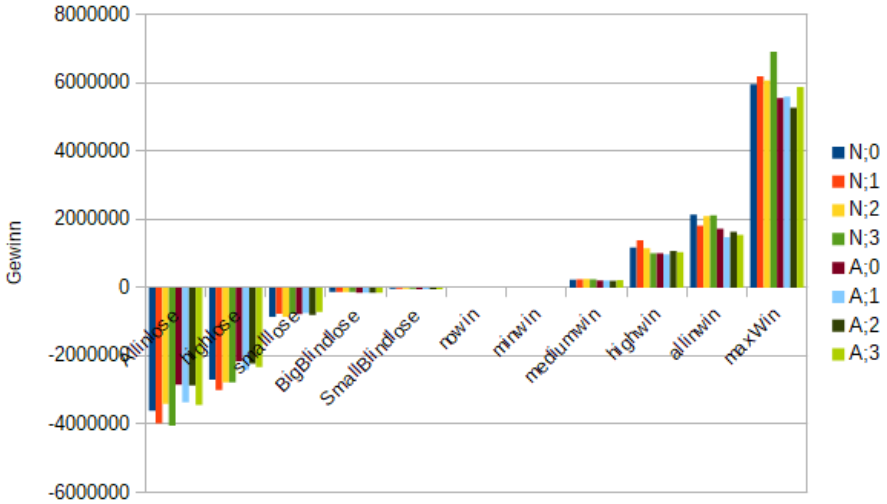


Abbildung 11: Grafische Darstellung der Gewinn-/Verlustverteilung der gemappten CFR-Varianten gegen zwei EV-Bots in Kategorien: Gewinne

- MinWin: Gewinn von 150 Chips, nachdem beide Gegner gefoldet haben.
- MediumWin: Gewinn zwischen 151 und 3000 Chips.
- HighWin: Gewinn zwischen 3001 und 19999 Chips.
- Allinwin: Gewinn zwischen 20000 und 39999 Chips.
- MaxWin: Gewinn von 40000 Chips was einem All-in von allen drei Spielern entspricht.

Zu beobachten ist, dass alle Varianten des Bots öfter verlieren als gewinnen. Am häufigsten verlieren die Bots den Small Blind, Big Blind oder den SmallLose. Was jedoch wie in Abbildung 11 zu sehen ist, einen sehr kleinen Verlust ausmacht. Viel höher ist der Verlust bei All-ins und bei Bigloses. Der höchsten Gewinne bekommen die Bots durch MaxAllin Gewinne.

Auffällig ist, dass alle Bot-Varianten, die die Potsize Anpassen, deutlich öfter im Big und im Small Blind folden. Dies liegt an den höheren Raise Werten, dadurch folded der Bot häufiger bei weniger guten Karten.

Tabelle 11: Win Lose Statistiken

	N;0	N;1	N;2	N;3	A;0	A;1	A;2	A;3
Lose	4886	4682	4889	4654	5067	4980	5084	4955
Win	642	657	652	674	591	587	869	592
AllIn Win%	0,568	0,538	0,588	0,567	0,604	0,554	0,591	0,559

In Tabelle 11 ist zu erkennen, dass die Potsize angepassten Bots deutlich häufiger verlieren, wobei es sich jedoch meistens um geringe Verluste handelt. Der Bot A;0, der den höchsten reinen Gewinnen erzielt hat, hat den höchsten All-in % Gewinn von über 60 Prozent. In Vergleich dazu haben N;1 und A;1, die Bot-Varianten mit dem geringsten Gewinnen, haben den niedrigsten All-in % Gewinn.

Ein Problem besteht in dem Realismus des EV Bots. So ist es sehr unwahrscheinlich, dass so häufig drei Spieler All-in gehen. Da jedoch die meisten Gewinne der Bots durch diese doppelten All-in Gewinne kommen, ist fraglich, ob die Ergebnisse nicht deutlich schlechter ausfallen würden, wenn die Bots weniger häufig 3-Way-All-ins(drei Spieler sind All-in) gewinnt.

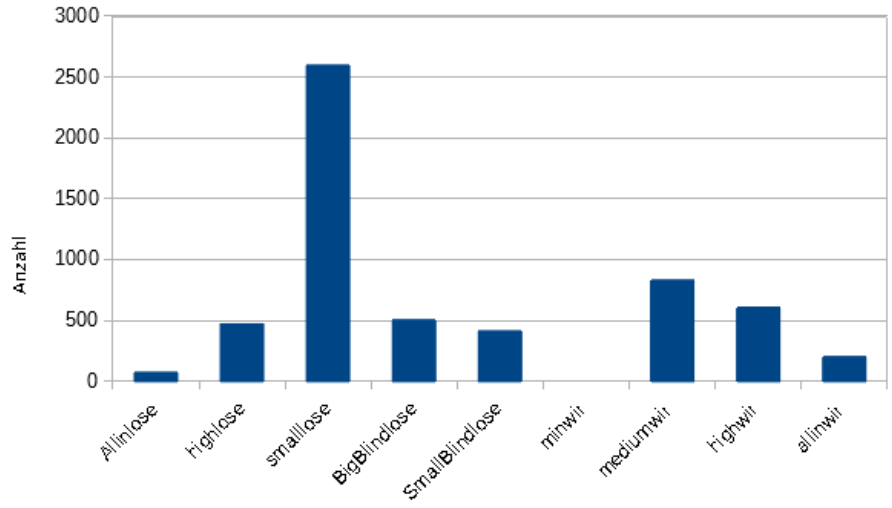


Abbildung 12: Grafische Darstellung von Gewinn-/Verlustverteilung des CFR gegen EVBot in Kategorien: Anzahl

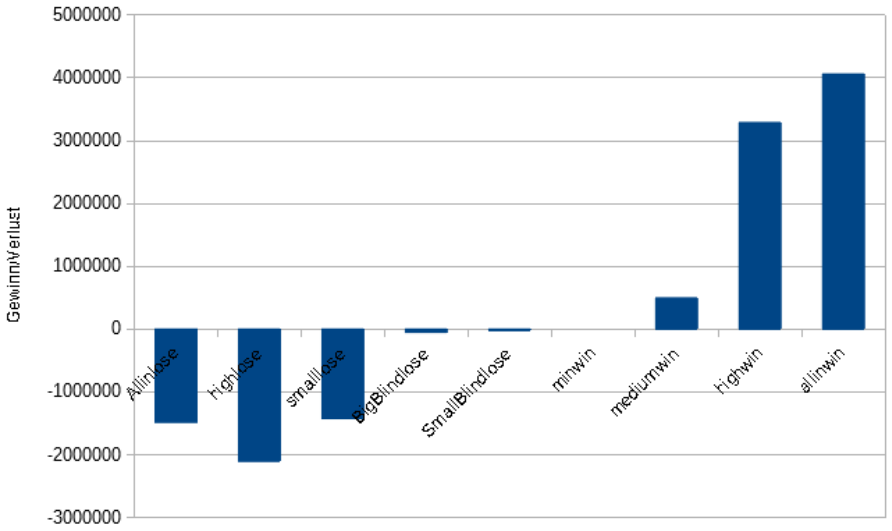


Abbildung 13: Grafische Darstellung der Gewinn-/Verlustverteilung des CFR Bots gegen einen EVBots in Kategorien: Gewinne

Im Vergleich dazu sieht die Gewinnverteilung des CFR Bots, dargestellt in Abbildung 12 und 13, ein wenig anders aus. Auffällig ist die All-in Winrate auf 73 Prozent steigt. Dies ist aber damit zu erklären, dass die Gewinnwahrscheinlichkeit in einem 3-Way All-in stark sinkt. So beträgt die All-in Equity von der Hand AhAs in einer Zwei-Spieler-Situation 84,5 %, in einer Drei-Spieler-Situation sinkt diese jedoch auf 74 % [3]. Generell gewinnt der CFRBot mit circa 1600 Händen deutlich mehr Hände als alle gemappten Varianten. Auch dieses Verhalten ist zum Teil auf die gesunkene Gewinnwahrscheinlichkeit in einem Drei-Spieler-Spiel zu zuschrieben, jedoch nicht in diesem Ausmaß. Dafür sind die Gewinne des gemappten Bots auch entsprechend höher. Weiterhin ist auffällig, dass die HighWins deutlich zunehmen, die Anzahl der All-ins jedoch abnimmt.

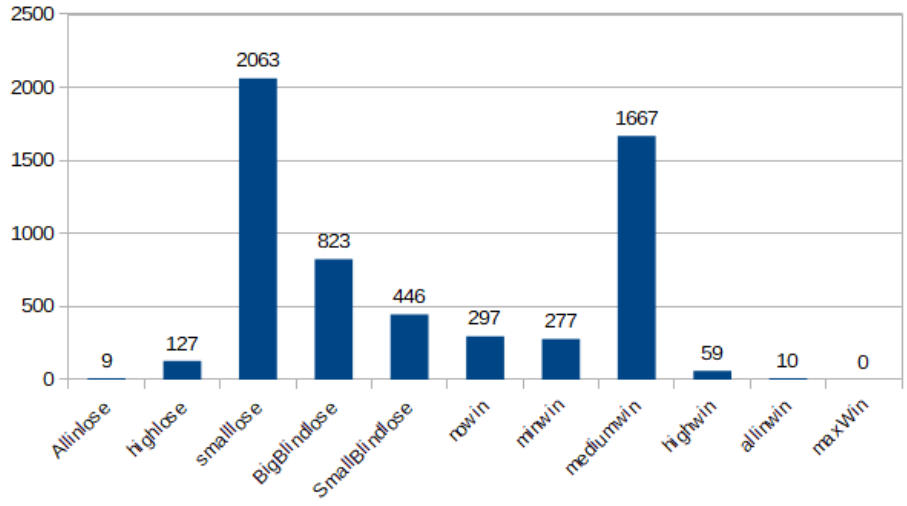


Abbildung 14: Grafische Darstellung von Gewinn-/Verlustverteilung des N;0 gegen A;3 EVBot in Kategorien

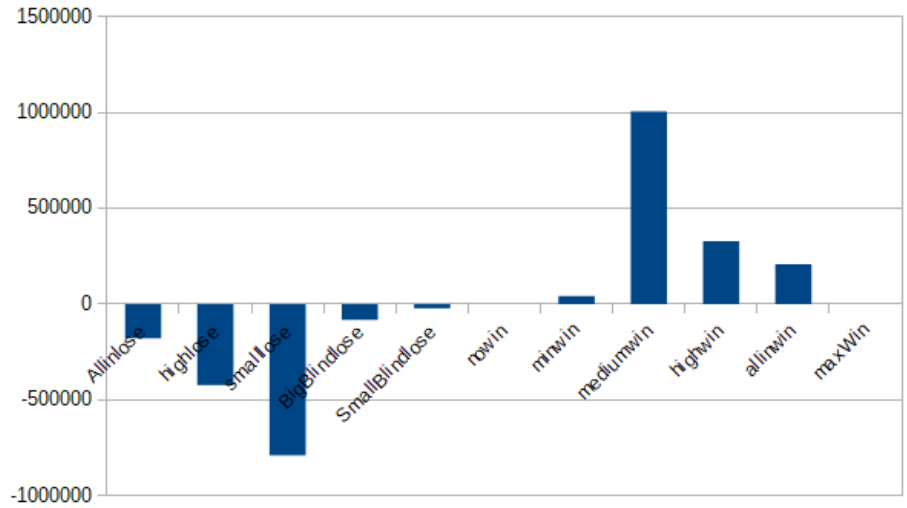


Abbildung 15: Grapfische Darstellung der Gewinn-/Verlustverteilung des CFR Bots gegen einen EVBots in Kategorien: Gewinne

In Abbildung 14 und 15 ist die Gewinnverteilung des N;0 gegen zwei A;3 Bots als Beispieldarstellung für die verschiedenen Bot Varianten. Interessant ist die Häufigkeit der All-ins. Im Gegensatz zu den Spielen gegen den EVBot ist die All-in Rate mit nur 19 All-ins sehr gering. Einen 3-Way-All-in gab es gar nicht. So kommen die höchsten Verluste durch SmallLoses und die höchsten Gewinne durch MediumWins zustande. Im Spiel gegeneinander scheinen die Bots so nicht sonderlich viel zu investieren.

6.3 Aktionshäufigkeiten

Bei den Aktionshäufigkeiten der unterschiedlichen Bots ist zu sehen, dass der CFR Bot sehr oft called oder folded während der EVBot fast nur raised und sehr selten folded oder called, was zu häufigen All-ins beider EVBots führt, da sie sich in einer Drei-Spieler-Situation häufig gegenseitig zum All-in bringen. Dieses Verhalten erklärt auch den anstieg der All-ins und die Verringerung der HighWins des CFR Bots im Vergleich zu den gemappten Varianten aus 6.2. Auch die gesunkene Hand Gewinnrate kann durch dieses Verhalten begründet werden, da durch die hohen Raises die gemappten Bots deutlich mehr Chips investieren müssen, was den Bot schneller folden lässt.

Tabelle 12: Aktionsverteilung A;0 gegen zwei EvBots

A;0	Anz	Anz%	Preflop	Flop	Trun	River
Fold	4980	43%	4637	340	3	0
Call	3722	32%	2120	712	878	887
Raise	3197	25%	2608	303	4	0
	11617					

Tabelle 13: Aktionsverteilung N;1 gegen zwei EvBots

N;1	Anz	Anz%	Preflop	Flop	Trun	River
Fold	4773	36%	4301	471	1	0
Call	5025	39%	3038	881	1122	1105
Raise	3197	25%	2815	377	5	0
	12995					

Wenn die verschiedenen gemappten Bot Varianten miteinander verglichen werden, fällt auf, dass alle Bots, bei denen die Potsize nicht angeglichen wurde, öfter callen als die Varianten bei denen die Potsize nicht angeglichen wurden. Zu erklären ist dieses Verhalten mit der Erhöhung des Amount-to-Call bei der Potsize Anpassung, was zu einer vorsichtigen Spielweise dieser Bots führt. Gegen den EVBot scheinen die passiven Bots besser abzuschneiden als die öfter callenden Bots. Post-PreFlop folden alle Varianten des CFR Bots kaum noch, spätestens beim Turn folden sie nur äußerst selten. So sind die größten Unterschiede der Varianten beim Preflop zu finden.

Tabelle 14: Aktionverteilung A;0 gegen zwei N;0 Bots

A;0	Anz	Anz%	Preflop	Flop	Turn	River
Fold	3174	25%	2663	487	24	0
Call	5889	47%	1755	2154	1937	1956
Raise	3393	27%	2478	911	4	0
	12456					

Tabelle 14 zeigt nun die Aktionsverteilung des A;0 gegen zwei N;0.Vergleicht man die Aktionhäufigkeit zwischen Tabelle 14 und Tabelle 12 ist zu erkennen, dass der Bot deutlich seltener folded, dafür deutlich häufiger called. Nur die Raiseanzahl hat sich prozentual gesehen kaum verändert. Dies zeigt, das die Bot-Varianten, wie auch in 6.2 vermutet deutlich weniger Chips von sich aus Investieren.

6.4 Aktionsvariation

Um die Unterschiede der verschiedenen Mapping Varianten aufzuzeigen, kann sich angeschaut werden wie oft die Bots verschiedene bzw. gleiche Aktionen tätigen. So unterscheiden sich z. B. der A;0 und der N;1, die Bots mit dem besten und schlechteren Ergebnis gegen den EVBot, in 13 % der Aktionen. Lässt man den A;0 gegen zwei N;1 Bots Spielen ergibt sich eine Action Differenz von 11 %.

Tabelle 15: Aktionsunterschiede des N;0 gegen jeweils zwei Bots der anderen gemmapten CFR-Varianten in Prozent

	N;0
N;1	1,4%
N;2	0,3%
N;3	1,3%
A;0	10,9%
A;1	11;3%
A;2	10,6%
A;3	11,0%

Tabelle 16: Aktionsunterschiede des N;0 gegen zwei EvBots mit Aktionsvergleich zu den anderen gemappten CFR-Varianten in Prozent

	N;0
N;1	8,2%
N;2	4,0%
N;3	8,0%
A;0	10,8%
A;1	12,6%
A;2	11,3%
A;3	12,8%

Betrachtet man nun Tabelle 15 und 16 ist deutlich zu erkennen, dass der Bot, wenn er gegen sich selber spielt, deutlich größere Unterschiede produziert, wenn die Potsitze angepasst wird. Diese Call-Varianten machen jedoch nur ein sehr kleinen Unterschied aus. Gegen den EVBot jedoch sorgen die verschieden Call-Varianten für ein deutlich größeren Unterschied. Dieses liegt an dem aggressiven Spiel der EVBots. Die Call-Varianten werden nur dann gebraucht wenn beide gegnerischen Bots oft raisen so wie der eigene Bot oft Called. Wie in 6.3 erwähnt raisen die EVBots sehr häufig und der CFR Called sehr häufig (Folden müssen nicht betrachtet werden, da dann auch kein Mapping mehr stattfindet). Dies sorgt dafür, dass die Call-Varianten deutlich häufiger angewendet werden müssen, was den größeren Einfluss auf das Mapping hat.

Auffällig ist, dass vor allem die Call Mods 0 und 2 sowie 1 und 3 kleinere Unterschiede aufweisen. Dies liegt logischerweise an der Implementierung von Call Mode 2 und 3, da wie in 5.3 beschrieben, bei nicht eintreten eines “CRR” Schemas Modus 0 bzw. 1 ausgeführt wird. Wenn die verschieden Varianten des Bots gegeneinander antreten ist auffällig, dass Unterschiede zwischen den Call Mods 0,2 so wie 1,3 kaum vorhanden sind, was dafür spricht, dass der Bot hier selten ein CRR auswertet.

6.5 Verluste durch das Mapping

Um einen Indikator zu bekommen, wie hoch die Verluste durch das Mapping sind, können die gemappten Spielzustände auch von dem EVBot ausgewertet werden. So können diese Ergebnisse mit den Ergebnissen des EVBot verglichen werden. In Tabelle 17 wird der durschnittliche Verlust von zwei EVBots mit gemappten Spielzuständen gegen einen N;0 Bot in bb/100 dargestellt. Der Verlust eines normalen EVBot bei dem Test gegen den N;0 betrag 169,29 bb/100. Generell ist der Verlust der gemappten EVBot um über 100 bb/100 höher als bei dem normalen EVBot. Die Varianten in denen die Potsize Angepasst werden, verlieren deutlich höher als die Varianten die die Potsize nicht Anpassen. Dies ist ein Indikator dafür, dass die Potsize Anpassung die und die damit verbundene Erhöhung der Amount-to-call die Pod Odds deutlicher verändert als die Nichtanpassung der Potsize.

Tabelle 17: Verluste Gemappter EVBot gegen N;0 in bb/100

	Verlust in bb/100
EV N;0	-287,42
EV N;1	-259,90
EV N;2	-290,60
EV N;3	-284,83
EV A;0	-404,17
EV A;1	-398,00
EV A;2	-395,00
EV A;3	-384,61

7 Fazit

Diese Thesis behandelte den Versuch einen Drei-Spieler-No-Limit-Texas-Hold'em Bot zu erstellen unter Verwendung eines spielstarken Zwei-Spieler-Bots. Dazu wird versucht eine Drei-Spieler-Situationen in eine Zwei-Spieler-Situationen zu mappen und mit dem gemappten Spielzustand, den Zwei-Spieler-Bot zu betreiben. Als Zwei-Spieler-Bot wurde ein auf dem Counterfactual Regret Minimization Algorithmus basierender Bot ausgewählt. Da es kaum vergleichbare Bots gibt ist die Stärke der Bots schwer zu vergleichen. Trotzdem wurde mit vorhandenen Möglichkeiten versucht die Stärke des Bots zu bewerten.

Zusammenfassend kann gesagt werden, dass das Mapping die minimalen Ziele erreicht hat und zumindestens die primitiven Bots wie AllinBots und minRaiseBots schlägt. Auch die Ergebnisse gegen den EVBot sehen nicht schlecht aus, wobei hier beachtet werden muss, dass die meisten Gewinne durch die 3-Way-All-ins zustande kommen, welche vor allem auf das Zusammenspiel der zwei EVBots zurückzuführen ist. Der Vergleich gegen Menschliche Spieler konnte nicht erbracht werden. Auch einen Vergleich gegen stärkere Drei-Spieler Bots konnte mangels passender Bots nicht erbracht werden.

Im Vergleich zum CFRBot sind Verluste zu erkennen, welche zu erwarten waren. Da die durch den CFR Algorithmus erstellte Nash Equilibrium auf ander Situationen trainiert wurde. Jedoch können einige Varianten beim Gewinn noch mit dem CFRBot mithalten.

Die verschiedenen Bot Varianten sind schwer miteinander zu vergleichen. So sind die Ergebnisse zwischen den verschiedenen Bots statistisch nicht relevant. Dies liegt vor allem daran, dass nachdem ein Spieler folded der weitere Verlauf durch den CFR bestimmt wird und somit die Unterschiede zwischen den Strategien sehr gering sind. Gegen den All-in Bot schließen alle Varianten ähnlich ab. Große Unterschiede gibt es vor allem gegen den EVBot. Hier schließt A;0 am besten ab. Das heißt jedoch nicht, dass dieser gegen andere Bots zwingend der Stärkste sein muss.

Abschließend sehen die Ergebnisse der verschiedenen Bots auf dem ersten Blick nicht schlecht aus und haben alle verfügbaren Bots deutlich geschlagen. Trotzdem sollte wenn möglich, noch weiter Tests durchgeführt werden. Als weiterer Ansatz könnten hier Daten aus realen Pokerspielen genutzt und so versucht werden Entscheidungen von realen Spielen mit den Entscheidungen des gemappten CFR zu vergleichen. Eine weitere Möglichkeit das Mapping weiter zu verbessern wäre eine Verbindung der verschieden Bot Varianten, um Stärken zu nutzen und Schwächen auszugleichen.

Literatur

[1] Annual Computer Poker Competition. Website: http://www.computerpokercompetition.org/. Zugriff am 30.12.2018.

[2] DeepStack Webseite. https://www.deepstack.ai/. Zugriff am 30.12.2018.

[3] Poker Odds Calculator. Abrufbar: https://www.poker.de/guides/wahrscheinlichkeiten-rechner/. Zugriff am 30.12.2018.

[4] TU-Darmsatdt Pokerframework. Abrufbar: https://www.ke.tu-darmstadt.de/svn/poker/pokertud/. Zugriff am 30.12.2018.

[5] Peter Glöckner. Abbildung eines 2-Spieler Computerspielers auf 3-Spieler-Limit-Poker, 2013.

[6] Michael Bradley Johanson. Robust Strategies and Counter-Strategies: Building a Champion Level Computer Poker Player. Master's thesis, University of Alberta, 2007.

[7] Suiteng Lu. Online enhancement of existing nash equilibrium poker agents. Master's thesis, University of Alberta, 2016.

[8] Lanctot Marc, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte carlo sampling for regret minimization in extensive games. In Advances in Neural Information Processing Systems 22, pages 1078–1086. 2009.

[9] Ed Miller. Getting Started in Hold'em. Two Plus Two Publishing, 2005.

[10] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker Poker Player. Science, 356(6337):508–513, 2017.

[11] Suiteng Lu Patryk Hoper. Computer Poker Praktikum WS 14/15. Technical report, TU Darmstadt, 2015.

[12] Nick Abou Risk and Duane Szafron. Using counterfactual regret minimization to create com- petitive multiplayer poker agents. Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, page 159–166, 2010.

[13] David Sklansky. The Theory of Poker. Two Plus Two Publishing, 2005.

[14] Oskari Tammelin, Neil Burch, Neil Burch, and Michael Bowling. Solving heads-up limit Texas Hold'em. Proceeding IJCAI'15 Proceedings of the 24th International Conference on Artificial Intelligence, pages 645–652, 2015.

[15] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. Advances in neural information processing systems, 20:1729–1736, 2008.

```
1       public int getAction(GameState gameState) {
            GameState twoPlayerState = null;
3           ArrayList<Player> pl = convertPlayer(gameState);// Spieler Anpassen
            if (direktFolded) {  // Sonderfall BU erste Aktion Fold
5               LinkedList<Integer> list = new LinkedList<Integer>();
                for (Integer i : gameState.getPreflopAction()) {
7                   list.add(i);
                }
9               list.removeFirst();
                twoPlayerState = new GameState(
11                      gameState.getRoundIndex(),
                        pl,
13                  gameState.getFlop().clone(),
                            gameState.getTurn().clone(),
15                          gameState.getRiver().clone(),
                            list,
17                          gameState.getFlopAction(),
                            gameState.getTurnAction(),
19                          gameState.getRiverAction());
            } else if (onePlayerHasFolded &&
21                  !(gameState.getCurrentStreet().equals(playerFoldedOnStreet))){
                realPotSize = 150;
23              if (playerFoldedOnStreet.equals(Street.PREFLOP)) { // Spieler Folded auf dem PreFlop
                    twoPlayerState = new GameState(gameState.getRoundIndex(), pl, gameState.
                        getFlop().clone(),
25                              gameState.getTurn().clone(), gameState.getRiver().clone(),
                                getConvertetActions(gameState.getPreflopAction(), Street.
                                    PREFLOP, gameState),
27                              gameState.getFlopAction(), gameState.getTurnAction(),
                                    gameState.getRiverAction());

29              } else if (playerFoldedOnStreet.equals(Street.FLOP)) {// Spieler Folded auf dem Flop
                    twoPlayerState = new GameState(gameState.getRoundIndex(), pl, gameState.
                        getFlop().clone(),
31                              gameState.getTurn().clone(), gameState.getRiver().clone(),
                                getConvertetActions(gameState.getPreflopAction(), Street.
                                    PREFLOP, gameState),
33                              getConvertetActions(gameState.getFlopAction(), Street.FLOP,
                                    gameState),
                                gameState.getTurnAction(), gameState.getRiverAction());
35
                } else if (playerFoldedOnStreet.equals(Street.TURN)) {// Spieler Folded auf dem turn
37                  twoPlayerState = new GameState(gameState.getRoundIndex(), pl, gameState.
                        getFlop().clone(),
                                gameState.getTurn().clone(), gameState.getRiver().clone(),
39                              getConvertetActions(gameState.getPreflopAction(), Street.
                                    PREFLOP, gameState),
                                getConvertetActions(gameState.getFlopAction(), Street.FLOP,
                                    gameState),
41                              getConvertetActions(gameState.getTurnAction(), Street.TURN,
                                    gameState),
                                gameState.getRiverAction());
43              }

45          } else { // Kein Spielerhat gefolded
                twoPlayerState = new GameState(gameState.getRoundIndex(), pl, gameState.getFlop().
                    clone(),
47                          gameState.getTurn().clone(), gameState.getRiver().clone(),
                            getConvertetActions(gameState.getPreflopAction(), Street.PREFLOP,
                                gameState),
49                          getConvertetActions(gameState.getFlopAction(), Street.FLOP, gameState)
                                ,
                            getConvertetActions((gameState.getTurnAction()), Street.TURN,
                                gameState),
51                          getConvertetActions((gameState.getRiverAction()), Street.RIVER,
                                gameState));//Anpassen der Actionshistorie

53          }
            twoPlayerState.setCurrentStreet(gameState.getCurrentStreet());
55          twoPlayerState.getPlayersContainer().setPositionToAct(twoPlayerState.getHeroPosition());
            twoPlayerState.setMaxBetsize(gameState.getMaxBetsize());
57          twoPlayerState.setPotsize(twoPlayerPotSize);

59          int action = bot.getAction(twoPlayerState); // CFR Bot nach Aktionfragen.
            return action;
61      }
        private LinkedList<Integer> getConvertetActions(LinkedList<Integer> lst, Street cs, GameState
            gameState) {
63          LinkedList<Integer> ret = new LinkedList<Integer>();

65          if (onePlayerHasFolded) {
                LinkedList<Integer> change = new LinkedList<Integer>();
67              LinkedList<Integer> folded = new LinkedList<Integer>();
                boolean fold = false;
69              for (Integer i : lst) {
                    if (fold) {
71                      folded.add(i);
                    } else {
73                      if (i == -1) {
                            fold = true;
75                      }
                        change.add(i);
77                  }
                }
79              ArrayList<LinkedList<Integer>> split = splitIntoThree(change,
                        getPositionToAct(gameState.getHeroPosition(), cs));
81              int pos = getPositionToAct(gameState.getHeroPosition(), cs);
                ret = convertActions(split, pos, cs, gameState);
```

```
83                          LinkedList<Integer> newFolded = new LinkedList<Integer>();
85                          int posFold = change.size() - 1;
                            int array[] = new int[2];
87                          array[0] = (posFold + 1) % 3;
                            array[1] = (posFold + 2) % 3;
89                          int count = 0;
                            for (int i : folded) {
91                                  if (i > 0) {
                                            invested[array[count % 2]] = i;
93                                          newFolded.add(i);
                                            twoPlayerPotSize = invested[pos] + Math.max(invested[(pos + 1) % 3],
                                                 invested[(pos + 2) % 3]);
95                                          realPotSize = invested[0] + invested[1] + invested[2];
                                            valueDiff = realPotSize - twoPlayerPotSize;
97                                  } else {
                                            invested[array[count % 2]] = invested[array[(count + 1) % 2]];
99                                          newFolded.add(i);
                                            twoPlayerPotSize = invested[pos] + Math.max(invested[(pos + 1) % 3],
                                                 invested[(pos + 2) % 3]);
101                                         realPotSize = invested[0] + invested[1] + invested[2];
                                            valueDiff = realPotSize - twoPlayerPotSize;
103                                 }
                                    count++;
105                         }
107                         ret.addAll(newFolded);
109                 } else {
                            ret = convertActions(splitIntoThree(lst, getPositionToAct(gameState.getHeroPosition(),
                                 cs)),
111                                         getPositionToAct(gameState.getHeroPosition(), cs), cs, gameState);
                    }
113                 if (valueMode == 1) {
                            ret = investedAnpassenEasy(ret, getPositionToAct(gameState.getHeroPosition(), cs),
                                 gameState);
115                 }
                    return ret;
117         }
119         private LinkedList<Integer> lookUp(LinkedList<Integer> lst, int pos) {
121                 LinkedList<Integer> ret = new LinkedList<Integer>();
                    if (lst.size() == 0) {// {}
123                         return ret;
                    }
125                 if (lst.size() == 1) {//
                            if (lst.get(0) == -1) {// {F}
127                                 return ret;
                            } else {// {R, F}
129                                 ret.add(lst.get(0));
                                    invested[(pos + 2) % 3] = lst.get(0);
131                                 return ret;
                            }
133                 } else {
                            if (lst.get(1) > 0) {
135                                 if (lst.get(0) > 0) {// RR)
                                            invested[(pos + 1) % 3] = lst.get(0);
137                                         invested[(pos + 2) % 3] = lst.get(1);
                                            ret.add(lst.get(1));
139                                         return ret;
                                    } else {// CR)
141
                                            invested[(pos + 1) % 3] = invested[pos];
143                                         invested[(pos + 2) % 3] = lst.get(1);
                                            ret.add(lst.get(1));
145                                         return ret;
                                    }
147                         } else if (lst.get(0) > 0) {// RC)
                                    invested[(pos + 1) % 3] = lst.get(0);
149                                 if (lst.get(1) != -1)
                                            invested[(pos + 2) % 3] = invested[(pos + 1) % 3];
151                                 ret.add(lst.get(0));
                                    return ret;
153                         } else {// CC
                                    invested[(pos + 1) % 3] = invested[pos];
155                                 if (lst.get(1) != -1)
                                            invested[(pos + 2) % 3] = invested[pos];
157                                 ret.add(0);
                                    return ret;
159                         }
161                 }
            }
163
            private LinkedList<Integer> convertActions(ArrayList<LinkedList<Integer>> arr, int pos, Street cs,
                        GameState gamestate) {
                    LinkedList<Integer> ret = new LinkedList<Integer>();
167
                    if (cs.equals(Street.PREFLOP)) {
169
                            invested[1] = 50;
171                         invested[2] = 100;
173                         twoPlayerPotSize = 150;
                            realPotSize = 150;
175                 }
                    int n = 0;
177                 valueDiff = 0;
                    if (pos == 0) {
179                         if (arr.size() == 0)
```

```
                        return ret;
                n = 1;
                if (arr.get(0).get(0) == 0 && cs.equals(Street.PREFLOP)) {
                        invested[0] = 100;

                }
                if (arr.get(0).get(0) > 0 && cs.equals(Street.PREFLOP)) {
                        invested[0] = arr.get(0).get(0);
                }

                twoPlayerPotSize = invested[pos] + Math.max(invested[(pos + 1) % 3], invested[(pos +
                    2) % 3]);
                realPotSize = invested[0] + invested[1] + invested[2];
                valueDiff = realPotSize - twoPlayerPotSize;
                ret.add(arr.get(0).get(0));
        }

        int counter = 0;
        for (int i = n; i < arr.size(); i++) {
                if (counter % 2 == 1) {
                        if (arr.get(i).get(0) == 0) {
                                if (arr.size() == i + 1) {
                                        ret.add(0);
                                        invested[pos] = Math.max(invested[(pos + 1) % 3], invested[(
                                            pos + 2) % 3]);
                                        twoPlayerPotSize = invested[pos] + Math.max(invested[(pos + 1)
                                             % 3], invested[(pos + 2) % 3]);
                                        realPotSize = invested[0] + invested[1] + invested[2];
                                        valueDiff = realPotSize - twoPlayerPotSize;
                                        return ret;
                                } else if (arr.get(i + 1).size() == 1) {
                                        ret.add(0);
                                        invested[pos] = Math.max(invested[(pos + 1) % 3], invested[(
                                            pos + 2) % 3]);
                                        if (arr.get(i + 1).get(0) == 0) {
                                                invested[(pos + 1) % 3] = Math.max(invested[(pos + 1)
                                                    % 3], invested[(pos + 2) % 3]);
                                        }
                                        twoPlayerPotSize = invested[pos] + Math.max(invested[(pos + 1)
                                             % 3], invested[(pos + 2) % 3]);
                                        realPotSize = invested[0] + invested[1] + invested[2];
                                        valueDiff = realPotSize - twoPlayerPotSize;
                                        return ret;
                                } else {
                                        int retSize = ret.size();
                                        invested[pos] = Math.max(invested[(pos + 1) % 3], invested[(
                                            pos + 2) % 3]);
                                        ret = actionByCall(i, arr, ret, mode, pos);

                                        twoPlayerPotSize = invested[pos] + Math.max(invested[(pos + 1)
                                             % 3], invested[(pos + 2) % 3]);
                                        realPotSize = invested[0] + invested[1] + invested[2];
                                        valueDiff = realPotSize - twoPlayerPotSize;
                                }

                        } else {

                                invested[pos] = arr.get(i).get(0);
                                ret.add(arr.get(i).get(0));
                                twoPlayerPotSize = invested[pos] + Math.max(invested[(pos + 1) % 3],
                                    invested[(pos + 2) % 3]);
                                realPotSize = invested[0] + invested[1] + invested[2];
                                valueDiff = realPotSize - twoPlayerPotSize;

                        }
                } else {
                        ret.addAll(lookUp(arr.get(i), pos));
                        twoPlayerPotSize = invested[pos] + Math.max(invested[(pos + 1) % 3], invested
                            [(pos + 2) % 3]);
                        realPotSize = invested[0] + invested[1] + invested[2];
                        valueDiff = realPotSize - twoPlayerPotSize;
                }
                counter++;
        }

        return ret;
    }
    private ArrayList<Player> convertPlayer(GameState gameState) {
        ArrayList<Player> players = gameState.getPlayers();
        Player hero = gameState.getHero().clone();
        Player opp = null;
        ArrayList<Player> ret = new ArrayList<Player>();

        if (onePlayerHasFolded) {
                if (gameState.getCurrentStreet().equals(Street.PREFLOP)) {
                        for (Player h : players) {
                                if (!h.hasFolded() && !h.isHero()) {
                                        opp = h.clone();
                                        if (h.getPosition().equals(Position.BB)) {
                                                // SB Folded
                                                if (hero.getPosition().equals(Position.BU)) {
                                                        hero.setPosition(Position.SB);
                                                }
                                        }
                                        //
                                        if (h.getPosition().equals(Position.SB)) {
                                                // BB folded
                                                if (hero.getPosition().equals(Position.BU)) {
                                                        hero.setPosition(Position.SB);
                                                        opp.setPosition(Position.BB);
                                                }
```

```
271                                         }
273                                         if (h.getPosition().equals(Position.BU)) {
                                                    // BB folded
275                                                 if (hero.getPosition().equals(Position.SB)) {
                                                            hero.setPosition(Position.BB);
277                                                         opp.setPosition(Position.SB);
                                                            // SB folded
279                                                 } else {
                                                            opp.setPosition(Position.SB);
281                                                 }
                                            }
283
                                    }
285                             }
                        } else {
287                             for (Player h : players) {
                                    if (!h.hasFolded() && !h.isHero()) {
289                                         opp = h.clone();
                                            if (h.getPosition().equals(Position.BB)) {
291                                                 // SB Folded
                                                    if (hero.getPosition().equals(Position.BU)) {
293                                                         hero.setPosition(Position.BB);
                                                            opp.setPosition(Position.SB);
295                                                 }
                                            }
297                                         if (h.getPosition().equals(Position.SB)) {
                                                    // BB folded
299                                                 if (hero.getPosition().equals(Position.BU)) {
                                                            hero.setPosition(Position.BB);
301                                                 }
                                            }
303
                                            if (h.getPosition().equals(Position.BU)) {
305                                                 // BB folded
                                                    if (hero.getPosition().equals(Position.SB)) {
307                                                         opp.setPosition(Position.BB);
                                                            // SB folded
309                                                 } else {
                                                            opp.setPosition(Position.BB);
311                                                         hero.setPosition(Position.SB);
                                                    }
313                                         }

315                                 }
                            }
317
                        }
319             } else {
                        if (gameState.getCurrentStreet().equals(Street.PREFLOP)) {
321                             int bb = 0;
                                if (hero.getPosition().equals(Position.SB)) {
323                                 for (Player h : players) {
                                        if (h.getPosition().equals(Position.BU)) {
325                                             opp = h.clone();
                                                opp.setPosition(Position.SB);
327                                             hero.setPosition(Position.BB);
                                        }
329                                     if (h.getPosition().equals(Position.BB)) {
                                            bb = h.getInvested();
331                                     }
                                    }
333                                 if (opp.getInvested() == 0) {
                                        opp.setInvested(bb / 2);
335                                     hero.setInvested(bb);
                                    }
337
                                }
339                             if (hero.getPosition().equals(Position.BB)) {
                                    for (Player h : players) {
341                                     if (h.getPosition().equals(Position.SB)) {
                                            opp = h.clone();
343                                     }
                                    }
345                                 // Player Postion = BU
                                } else {
347                                 for (Player h : players) {
                                        if (h.getPosition().equals(Position.BB)) {
349                                             hero.setPosition(Position.SB);
                                                opp = h.clone();
351                                     }
                                    }
353                                 if (hero.getInvested() == 0) {
                                        hero.setInvested(opp.getInvested() / 2);
355                                 }
                                }
357
                        } else {
359                             if (hero.getPosition().equals(Position.BB)) {
                                    for (Player h : players) {
361                                     if (h.getPosition().equals(Position.SB)) {
                                            opp = h.clone();
363                                     }
                                    }
365                             }
                                if (hero.getPosition().equals(Position.SB)) {
367                                 for (Player h : players) {
                                        if (h.getPosition().equals(Position.BB)) {
369                                             opp = h.clone();
                                        }
371                                 }
```

```
                                        // Player Postion = BU
373                                 } else {
                                        for (Player h : players) {
375                                         if (h.getPosition().equals(Position.SB)) {
                                                hero.setPosition(Position.BB);
377                                             opp = h.clone();
                                            }
379                                     }

381                                 }
                                }
383             }
                ret.add(hero);
385             ret.add(opp);

387             return ret;

389     }
```