Abstraktionsansätze in Multiplayer Poker

Komplexitätsanalyse und Abstraktionsmöglichkeiten des Setzbaumes in Multiplayer Fixed Limit Poker

Bachelor-Thesis von Jan Simon Bunten aus Schwäbisch Hall
Tag der Einreichung:

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

TECHNISCHE UNIVERSITÄT DARMSTADT

Informatik
Knowledge Engineering

Abstraktionsansätze in Multiplayer Poker
Komplexitätsanalyse und Abstraktionsmöglichkeiten des Setzbaumes in Multiplayer Fixed Limit Poker

Vorgelegte Bachelor-Thesis von Jan Simon Bunten aus Schwäbisch Hall

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

Tag der Einreichung:

Zusammenfassung

Multiplayer-Fixed-Limit Texas Hold'em kann von explorativen Algorithmen aufgrund seiner Menge an Zuständen nur mit groben Kartenabstraktionen berechnet werden. So grob, dass dabei nur unterschieden werden kann, ob die aktuelle Hand eine über- oder unterdurchschnittliche Gewinnwahrscheinlichkeit hat. Eine weitere Reduktion der Komplexität kann durch Kartenabstraktionen nicht erreicht werden. Deshalb evaluiere ich in dieser Arbeit Möglichkeiten, den Zustandsraum durch das Verringern der Anzahl von Knoten im Setzbaum zu reduzieren. Die Voraussetzung dafür ist eine umfassende Komplexitätsanalyse des 3-Spieler-Spielbaumes. Des Weiteren untersuche ich eine mögliche Reduktion des Zustandsraumes durch die Beschränkung des Lösungsalgorithmus auf 2-Spieler-Zustände und das Zusammenfassen von gleichen Spielzuständen nach dem Ende einer Setzrunde.

Zusammenfassung

Multiplayer Fixed Limit Texas Hold'em can only be solved with very course card abstractions due to its large number of Game States. In fact so course, that usually hands can only be distinguished by above or below average hand strength, so reducing the complexity by card abstractions alone is not possible. Because of that I evaluate in this thesis possibilities to reduce complexity by reducing the betting nodes in the game tree. Required for this is an in depth analysis of the 3-player game tree. Furthermore I evaluate a possible reduction of the state space by restricting the algorithm so solve states with 2 active players only and by merging similar states after the end of a betting round.

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 25. Januar 2017

(Jan Simon Bunten)

Inhaltsverzeichnis

1 Einleitung

Im Bereich der künstlichen Intelligenz - im nachfolgenden Text als KI bezeichnet - spielt die Untersuchung von Spielen für die Bewertung von Algorithmen seit langem eine bedeutende Rolle. Spiele eignen sich aus einigen Gründen gut für den Test und den Vergleich von Algorithmen:

- Klare Regeln – Spiele haben wohldefinierte Regeln, nach denen das Spiel gespielt wird.
- Vollständig beobachtbar - Der Spielzustand ist beobachtbar und unterschiedliche Zustände sind klar voneinander zu unterscheiden.
- Ergebnisse überpüfbar – Am Ende des Spieles gibt es eine unmittelbare Bewertung, wie gut der Algorithums das Problem gelöst hat.
- Experten vorhanden – Es gibt Experten, die als Vergleich für die Ergebnisse der Agenten herangezogen werden können.

Spiele haben eine Teilmenge von Eigenschaften, die sich auch bei Applikationen in der realen Welt finden. So kann unter kontrollierten Bedingungen überprüft werden, wie gut Agenten bestimmte Teilaspekte lösen können und damit, welche Agenten für welche Problemstellungen infrage kommen. Poker hat als Spiel einige besondere Eigenschaften, die es als Untersuchungsobjekt für die KI interessant macht. Ist das Ergebnis von Aktionen der Spieler immer eindeutig vorhersagbar und verändert sich der Zustand des Spiels alleine durch die Aktionen der Spieler, so spricht man von einem deterministischen Spiel. Gibt es Zufallsereignisse, die den Spielzustand oder die möglichen Aktionen eines Spielers beeinflussen, kennt der Agent bei der Planung seines Zuges die tatsächlich möglichen Aktionen seines Gegenspielers nicht. Poker hat durch das zufällige Austeilen von Karten an die Spieler eine stochastische Komponente.
Kann ein Spieler den gesamten Zustand des Spieles beobachten, besitzt ein Spiel vollständige Informationen. Beispiele hierfür sind z. B. Schach oder 4-Gewinnt. Gibt es hingegen Teile des Spieles, die von Spielern nicht beobachtet werden können, z. B. die Karten auf der Hand des Gegenspielers, handelt es sich um ein Spiel mit unvollständigen Informationen. Beim Poker sind dem Gegenspieler zumindest Teile der Karten seiner Gegenspieler unbekannt. Es handelt sich also um ein Spiel mit unvollständigen Informationen.
Poker kann mit mehr als zwei Spielern gespielt werden. Das hat Auswirkungen auf die Existenz einer optimalen Strategie.
Der Spieler kann versuchen, andere Spieler zu täuschen. Das Spielergebnis hängt nicht alleine von der Stärke der Karten der einzelnen Spieler ab. Spieler können durch hohe Einsätze eine starke Hand andeuten, um andere Spieler zur Aufgabe zu bewegen, obwohl diese eine bessere Hand halten.
Der Gewinner des Spieles ist der Spieler, der am Ende des Spieles den größten Gewinn erzielt hat. Es besteht jedoch die Möglichkeit, mit einer geringeren Anzahl an gewonnenen Runden einen höheren Gewinn zu erzielen als die Gegner, wenn die Gewinnsumme in den gewonnenen Runden entsprechend hoch ausfällt. Die Anzahl der gewonnen Runden ist nicht ausschlaggebend. Das Risiko zwischen hohen Einsätzen und hohen Gewinnen bzw. Verlusten muss immer abgewogen werden.

1.1 Motivation

In letzter Zeit erhalten KIs für die Lösung von Spielen wieder erhöhte Aufmerksamkeit. Mit dem ersten Sieg einer KI über einen Profi im sehr komplexen Brettspiel Go [Dön] sind die KIs in eine Domäne vorgedrungen, in der man das aufgrund der Komplexität des Spieles lange Zeit für unmöglich gehalten hatte. Auch im Bereich von Poker hat eine KI von sich reden gemacht, als die University of Alberta die Variante

2-Spieler-Fixed-Limit-Texas-Holdem quasi gebrochen hat. Sie konnten eine Strategie berechnen, gegen die ein Spieler, der die Best-Response-Strategie spielt, im Schnitt nicht mehr als 0.986 mbb/g (Milli-Big-Blind pro Spiel) gewinnt. D. h., der Gegner kann im besten Fall einen Big Blind in 1000 Spielen gewinnen, wenn die Spieler eine ausreichend große Zahl an Spielen gegeneinander spielen. Die Best Response Strategie ist dabei die Strategie, die gegen die untersuchte Strategie immer den Zug mit dem höchsten erwarteten Nutzen wählt, wobei der Best Response Spieler weiß, welche Züge der Gegenspieler jeweils tätigen wird. Die verwendete Technik, die Minimierung des counterfactual Regret (CFR), wird schon länger in spielstarken Agenten eingesetzt.
Diese Poker-Variante ist jedoch deutlich weniger komplex als die Varianten, die üblicherweise von Menschen gespielt werden. In klassischen Poker-Turnieren sitzen bis zu 10 Personen an einem Poker-Tisch und die No-Limit-Varianten erfreuen sich größter Beliebtheit. Beide Veränderungen des Spiels erhöhen die Komplexität um ein Vielfaches, was eine exakte Lösung durch explorative Algorithmen erschwert. Ein Algorithmus wie CFR wird ohne zusätzliche Abstraktionen daher sehr viel mehr CPU-Zeit und Speicher benötigen, um zu einer vergleichbar starken Strategie zu konvergieren. Außerdem gibt es für Mehrspieler-Varianten keine theoretischen Garantien für die Existenz einer nash-optimalen Strategie [Ris, S. 20-21].
Die Größe des Spielbaumes des 3-Spieler-Spieles ist unabstrahiert mit $\sim 5 * 10^{17}$ Information Sets ungefähr um den Faktor 1000 größer, als die 2-Spieler-Variante. Eine häufig verwendete Methode, das Spiel zu abstrahieren, ist das sogennante Card Bucketing, wobei Kartenkombinationen in Eimern (Buckets) zusammengefasst und mit der gleichen Strategie gespielt werden. Um eine Strategie für das 3-Spieler-Spiel zu berechnen, die so viel Speicherplatz wie eine 2-Spieler-Strategie benötigt, müsste man also die Anzahl der Buckets um den Faktor 1000 reduzieren. Das bedeutet einen immensen Informationsverlust und einen entsprechend großen Fehler, der durch die Abstraktion des Spieles entsteht. Eine 3-Spieler Strategie kann alleine mit Hilfe von Card Bucketing zwar gerade noch berechnet werden, aber die Anzahl von Kartenkombinationen, die vom Spieler unterschieden werden, wird dabei sehr gering und der Fehler, der durch die Abstraktion ensteht, entsprechend groß. Ich möchte daher andere Möglichkeiten untersuchen, um den Spielbaum zu abstrahieren. Dadurch könnte die Zahl an unterscheidbaren Kartenkombinationen in der 3-Spieler Strategie erhöht werden und sogar eine Strategie für mehr als drei Spieler mit Hilfe von CFR berechenbar werden. Da es in letzter Zeit wenige Neuerungen auf diesem Gebiet gab, wurde diese Spielvariante 2016 [Coma] sogar aus der Annnual Computer Poker Competition zugunsten der kleineren Variante Kuhnpoker entfernt, und bei der geplanten Competition für 2017 [Comb] gibt es keine Multiplayer-Variante mehr. Ich hoffe daher, dass ich mit dieser Arbeit einen Anstoß zu weiteren Entwicklungen im Bereich von Multiplayer Fixed-Limit Poker geben kann.

1.2 Ziel dieser Arbeit

Im Rahmen dieser Arbeit sollen Möglichkeiten evaluiert werden, den Zustandsraum von Multiplayer Fixed-Limit Poker zusätzlich zu Card-Bucketing-Techniken zu verkleinern. Dafür soll eine Komplexitätsanalyse von 3-Spieler-Fixed-Limit-Poker durchgeführt werden und die Effektivität von zwei möglichen Varianten damit überprüft werden.
In Multiplayer Fixed-Limit Poker ist es häufig der Fall, dass einer der Spieler bereits in einer der ersten Runden aus dem Spiel ausscheidet und das Spiel nur zwischen 2 Spielern zu Ende gespielt wird. Viele der möglichen Pfade, in denen noch alle Spieler im Spiel verbleiben, sind dagegen verhältnismäßig selten, da Spieler mit einer schlechten Starthand ohne bzw. mit geringen Verlusten aus dem Spiel aussteigen können. Daher soll die Möglichkeit evaluiert werden, nur Strategien für Teilbäume zu berechnen, in denen alle bis auf 2 Spieler aus dem Spiel ausgestiegen sind. Die übrigen Entscheidungen sollen später in einem Expertenteam von anderen Agenten übernommen werden.
Die Analyse des Spielbaumes hat gezeigt, dass sich die Größe des Spielbaumes zusätzlich zu Card-Bucketing-Techniken am effektivsten durch die Reduktion der Knoten, mit denen spätere Setzrunden beginnen, verkleinern lässt. Bei der zweiten Variante werde ich deshalb darlegen, dass gleiche Spielzu-

stände am Ende einer Setzrunde für den Beginn der nächsten Runde zu einem gemeinsamen Knoten zusammengefasst werden können, um den Branching-Faktor des Spielbaumes zu verringern. Außerdem werde ich zeigen, dass diese Abstraktion des Spielbaumes bei steigenden Spielerzahlen effizient einsetzbar ist und der Grad der Abstraktion varriiert werden kann.

1.3 Aufbau dieser Arbeit

Kapitel 2 behandelt die notwendigen Vorkenntnisse für das Verständnis der Arbeit, die Regeln der hier untersuchten Poker-Variante, die mathematischen Grundlagen für den momentan erfolgreichsten Lösungsalgorithmus und die Komplexitätsanalyse für das 2-Spieler-Spiel.
In Kapitel 3 folgt die Komplexitätsanalyse des 3-Spieler-Spiels, in der die möglichen Biet-Sequenzen untersucht werden und daraus die Zahl der Knoten des Spielbaums hergeleitet wird.
In Kapitel 4 wird die Möglichkeit beschrieben, mit dem CFR-Algorithmus nur die 2-Spieler-Knoten zu lösen. Und ich zeige auf, welche Garantien des CFR-Algorithmus in dieser Variante im Gegensatz zum normalen 3-Spieler-Fall erhalten bleiben.
Kapitel 5 behandelt einen Ansatz, gleiche Spielzustände nach dem Ende einer Setzrunde zu einem gemeinsamen Knoten zusammenzufassen, um den Branching-Faktor des Baumes zu reduzieren.
In Kapitel 6 folgt ein Ausblick, wie sich bei beiden betrachteten Varianten die Faktoren, um die sich die Komplexität des Baumes verringert, in der 4-Spieler-Variante entwickeln, um abzuschätzen, ob sie sich für größere Varianten des Spieles als Abstraktionen eignen.
In Kapitel 7 stelle ich gegenüber, wie sich die Knotentypen bei steigender Spielerzahl verändern.
In Kapitel 8 folgt ein Fazit über die untersuchten Ansätze und Anknüpfungspunkte für Folgearbeiten.

2 Grundlagen

2.1 Spielregeln und Terminologie von Fixed Limit-Texas Hold'em Poker

Die in dieser Arbeit untersuchte Poker Variante ist Fixed-Limit Texas Hold'em. In den anderen Teilen der Arbeit wird die Variante verkürzt als Fixed-Limit Poker bezeichnet. Da es nicht für alle Terminologien des Poker-Spiels eine deutsche Entsprechung gibt, werden die Bezeichnungen, bis auf die Namen der Karten und gängigen Begriffen aus Kartenspielen, aus dem Englischen übernommen. Die Informationen über die Regeln von Texas Hold'em entstammen Wikipedia [Hol].

Das verwendete Blatt besteht aus 52 Karten in 4 verschiedenen Farben (Karo, Pik, Herz, Kreuz). In jeder Farbe gibt es die gleichen Karten mit den Wertigkeiten Ass, König, Dame, Bube bzw. den Zahlenwerten 10 bis 2 in dieser Reihenfolge. Bei den Farben gibt es keine Ordnung.

Die möglichen Kartenkombinationen sind nach Stärke aufsteigend:

- High Card – höchste einzelne Karte
- Pair – 2 Karten mit gleichem Wert
- Two Pair – 2 unterschiedliche Pairs
- Three of a kind – 3 Karten mit gleichem Wert
- Straight – 5 Karten mit aufeinander folgendem Wert
- Full House – Three of a kind und Pair
- Flush – 5 Karten der gleichen Farbe
- Four of a kind – 4 Karten mit dem gleichen Wert
- Straight Flush – Straight, deren Karten alle die gleiche Farbe haben
- Royal Flush – Straight Flush mit den Kartenwerten 10 bis Ass

Eine dieser Kombinationen wird als ein Blatt bezeichnet.

Bei Texas Hold'em besteht die Hand der Spieler aus der Kombination von Community Cards und Hole Cards. Community Cards sind die insgesamt 5 Karten, die im Verlauf des Spiels ausgelegt werden und von allen Spielern genutzt werden, um eine möglichst starke Hand zu bilden. Hole Cards bezeichnen die beiden Karten, die die Spieler zu Beginn des Spieles erhalten und die nur ihnen bekannt sind. Das Blatt des Spielers am Ende des Spiels ist das beste Blatt aus 5 Karten, das aus den 7 Hole und Community Cards gebildet werden kann. Im Gegensatz zu anderen Varianten von Poker sind dem Spieler nur ein kleiner Teil der möglichen Hand seines Gegners unbekannt. Daher gilt Texas Hold'em als eine Variante, bei der das Glück eine geringere Rolle spielt als in anderen Poker Varianten.
Das Spiel besteht aus 4 Setzrunden, in denen die Spieler Einstätze aus ihrem Vorrat, dem Stack, ins Spiel bringen können. Die einzelnen Runden werden als Preflop, Flop, Turn und River bezeichnet. Vor dem Preflop müssen bereits eine Anzahl der Spieler einen Einsatz bringen. Bei diesen Einsätzen wird zwischen den zwei Varianten Blinds und Antes unterschieden. Blinds sind Einsätze von 2 Spielern in unterschiedlicher Höhe, Small Blind und Big Blind. Der Small Blind beträgt üblicherweise die Hälfte des Big

Blinds. Antes sind Zwangseinsätze, die alle Spieler vor der Runde in der gleichen Höhe einsetzen müssen.

Vor jeder der Setzrunden werden neue Karten an die Spieler ausgeteilt. Die Hole Cards vor dem Preflop, 3 Community Cards vor dem Flop und je eine Community Card vor dem River.

Die Besonderheit der Fixed-Limit-Variante besteht darin, dass die Einsätze der Spieler in Höhe und Anzahl beschränkt sind. Die Höhe wird festgelegt als Small Bet für Preflop und Flop und als Big Bet für Turn und River. In jeder Setzrunde können maximal 4 Bets eingesetzt werden und jede Erhöhung beträgt genau eine Bet der jeweiligen Runde.

In jeder der 4 Runden haben die Spieler 5 mögliche Aktionen:

- Bet: Einen Einsatz bringen, wenn noch kein Spieler einen Einsatz gebracht hat.
- Raise: Den Einsatz um einen bestimmten Betrag zu erhöhen.
- Call: Eine vorangegangene Erhöhung um den gleichen Betrag mitzugehen.
- Check: Den eingesetzten Betrag nicht weiter zu erhöhen, wenn der Spieler bereits so viel investiert hat wie der Meistbietende.
- Fold: Einen Einsatz nicht mitzugehen und aus der Runde auszusteigen.

Eine Runde endet, wenn alle Spieler mindestens einmal an der Reihe waren und wenn alle Spieler den gleichen Betrag eingesetzt haben oder aus dem Spiel ausgestiegen sind. Sind nach der letzten Runde, dem River, immer noch mehrere Spieler im Spiel, kommt es zum Showdown. Die Spieler müssen ihre Hole Cards aufdecken und die Blätter der Spieler werden verglichen, um den Gewinner zu ermitteln. Bei einem Unentschieden zwischen mehreren Spielern teilen sie die Einsätze aller Spieler, den Pot, zu gleichen Teilen unter sich auf.

2.2 Spiele in Normalform und extensiver Form

Die Normalform für Spiele ist eine häufig angewandte Formalisierung. Sie sind geeignet für Spiele, in denen einzelne Aktionen gleichzeitig von mehreren Agenten ausgewählt werden. Formal kann man ein Spiel in der Normalform wie folgt definieren:
Definition: Ein endliches Spiel in der Normalform G ist ein Tupel (N, A, u), das aus den folgenden Komponenten besteht:

- Einer Menge Spielern $N = \{1, 2, \cdots, n\}$
- Einer Menge von Aktionsprofilen $A = A_1 \times A_2 \times \cdots \times A_n$ wobei A_i eine endliche Menge von Aktionen ist, die Spieler i im Spiel G wählen kann.
- Eine Nutzenfunktion $u_i : A \to \mathbb{R}$ für alle $i \in N$, die den Nutzen von Spieler i für jedem möglichen Aktionsprofil A eine Auszahlung zuordnet.

Von einem Nullsummenspiel oder Zero-sum-Spiel sprechen wir, wenn das Spiel von nur zwei Spielern gespielt wird und wenn für deren Nutzenfunktionen $u_1 = -u_2$ gilt. Gelten diese Voraussetzungen nicht, spricht man von einem Non-zero-sum-Spiel.
Eine Strategie σ_i für einen Spieler $i \in N$ ist eine Wahrscheinlichkeitsverteilung über A_i, wobei $\sigma_i(a)$ jeder möglichen Aktion $a \in A_i$ eine Wahrscheinlichkeit zuordnet, mit der Spieler i die Aktion im Spiel auswählt. Betrachten wir als Beispiel das Spiel Schere, Stein, Papier. Eine mögliche Strategie für Spieler

1 wäre, jeweils in der Hälfte aller Spiele Stein oder Papier zu wählen und niemals Schere. Die entsprechende Strategie wäre dann:

$$\sigma_1(a) = \begin{cases} \frac{1}{2} \text{ wenn } a = \text{Stein} \\ \frac{1}{2} \text{ wenn } a = \text{Papier} \\ 0 \text{ wenn } a = \text{Schere} \end{cases} \tag{2.1}$$

Die Menge aller möglichen Strategien für den Spieler i ist definiert als Σ_i
Dabei sprechen wir von einer Pure Strategy, wenn der Spieler sich immer für die gleiche Aktion entscheidet, also eine der Aktionen mit der Wahrscheinlichkeit 1 auswählt. Ansonsten spricht man von einer mixed strategy. Im Beispiel oben handelt es sich also um eine mixed strategy. Eine Pure Strategy ist eine Strategie $s_i \in S_i$, wobei S_i analog zu Σ_i die Menge aller möglichen pure strategies beschreibt. Eine Menge von Strategien $\sigma = (\sigma_1, \sigma_2, \cdots, \sigma_n)$, die für jeden Spieler $i \in N$ eine zugehörige Strategie σ_i enthält, bezeichnet man als ein Strategieprofil $\sigma \in \Sigma$. Wollen wir aus einem Strategieprofil eine bestimmte Strategie entfernen, so bezeichnen wir das daraus resultierende Profil als σ_{-i}. Die Nutzenfunktion $u_i(\sigma)$ bezeichnet den Nutzen des Spielers, wenn er seine Entscheidungen nach dem Strategieprofil σ auswählt.
Neben der Normalform als Spieldefinition, die weit verbreitet genutzt wird, gibt es noch eine weitere Form, in der ein Spiel modelliert werden kann, die extensive Form. Spielemodellierungen in der extensiven Form werden bevorzugt verwendet wenn ein Spiel eine der folgenden Eigenschaften hat:

- Die Entscheidungen im Spiel werden sequenziell statt gleichzeitig getroffen.
- Die Spieler besitzen unvollständige Informationen über das Spiel.
- Aktionen im Spiel beruhen auf Wahrscheinlichkeiten.

Die Anwendung der extensiven Form ist nicht auf Spiele mit diesen Eigenschaften beschränkt. Es ist durchaus möglich, auch andere Spiele in der extensiven Form zu formulieren. Diese Darstellungsweise ist nur weniger üblich. In Fixed-Limit Poker sind alle diese Eigenschaften vorhanden, weshalb hier häufig die extensive Form für die Modellierung des Spiels verwendet wird. Ein Spiel in der extensiven Form wird durch einen gewurzelten, gerichteten Graphen dargestellt, bei dem die Knoten Entscheidungspunkte für die Spieler symbolisieren und die Kanten die gewählten Aktionen [Gib14, S. 7].
Wie auch bei Spielen in der Normalform wird bei Spielen unterschieden, ob ein Spiel zero-sum oder non-zero-sum ist. Die Voraussetzung für ein Zero-sum-Spiel ist, dass das Spiel nur zwischen 2 Spielern gespielt wird und für die utility function muss $u_1 = -u_2$ gelten. Dass heißt der Verlust des einen Spielers ist genau der Gewinn des anderen. Für jeden Spieler sind die Entscheidungsknoten zu Information Sets zusammengefasst, sodass die Entscheidungsknoten innerhalb der Information Sets für den Spieler nicht unterscheidbar sind. Information sets, die mehr als einen Zustand enthalten, entstehen durch Informationen, die nur für einen Teil der Spieler sichtbar sind, wie z. B. die Handkarten im Poker. Die formale Definition von extensive form games ist aus dem Buch von Osbourne und Rubinstein [OR94, S. 200] übernommen:
Ein endliches extensives Spiel mit unvollständiger Information besteht aus den folgenden Komponenten:

- Einer endliche Menge von Spielern $N = \{1, 2, ...n\}$
- Einer endliche Menge von Sequenzen H, die möglichen Historien der Spielaktionen, sodass H die leere Sequenz enthält und jeder Präfix h einer Sequenz $h' \in H$, geschrieben $h \sqsubseteq h'$ selbst auch in H enthalten ist. Ergänzend bezeichnet die Menge $Z \subseteq H$ die Terminalhistorien (alle Historien, die keine Präfixe einer anderen Historie sein können, also Historien, nach denen das Spiel beendet ist).
- Einer Spielerfunktion P, die jeder nichtterminalen Historie $h \in H \setminus Z$ ein Element $P(h) \in N \cup c$ zuordnet. $P(h)$ ist der Spieler, der nach der Historie h als Nächster an der Reihe ist eine Aktion zu wählen. Wenn $P(h) = c$, bestimmen Wahrscheinlichkeiten die Aktion nach der Historie h. Wir legen $H_i = \{h \in H \mid P(h) = i\}$ als die Menge von Historien, die Spieler i zugehören, fest.

- Einer Funktion σ_i, die jeder Historie in H_i des Spielers i eine Wahrscheinlichkeit $\sigma_i(h, \cdot)$ für $A(h)$ zuordnet, wobei alle Wahrscheinlichkeiten voneinander unabhängig sind. Für $h \in H_c$ und $a \in A(h)$ ist $\sigma_c(h, a)$ die Wahrscheinlichkeit, dass die Aktion a nach der Historie h eintritt.

- Einer Nutzenfunktion $u_i : Z \to \mathbb{R}$ für jeden Spieler $i \in N$, die die Auszahlung für Spieler i nach jeder möglichen Terminalhistorie angibt. Wir legen $\triangle_i = max_{z,z' \in Z} u_i(z) - u_i(z')$ als den Bereich des Nutzens für Spieler i fest.

- Einer Informationspartition $\mathscr{I}_i$ von H_i mit der Eigenschaft, dass $A(h) = A(h')$ immer dann, wenn h und h' sich im gleichen Element der Partition befinden. Jedes Element der Partition $I \in \mathscr{I}_i$ bezeichnen wir als ein Information Set. Wir bezeichnen $A(I)$ als die Menge von Aktionen $A(h)$ und $P(I)$ als den Spieler $P(h)$ für jedes beliebige $h \in I$. Zusätzlich bezeichne $I(h)$ für alle $h \in H_i$ das Information Set das h enthält. Außerdem bezeichne $|A(\mathscr{I}_i)| = max_{I \in \mathscr{I}} |A(I)|$ die maximal verfügbare Anzahl von Aktionen von Spieler i in einem beliebigen Information Set.

Um sich die Einteilung von Entscheidungsknoten in Information Sets zu verdeutlichen, kann man sich den Beginn eines Spieles vorstellen, bei dem der Spieler 2 Asse hält, aber die Karten der anderen Spieler für ihn unbekannt sind. Für den Spieler sehen alle Spiele mit dieser Hand gleich aus. Er kann nicht zwischen den Spielen unterscheiden, auch wenn seine Gegner unterschiedliche Hände halten. Deshalb wären alle Spiele, bei denen der Spieler mit 2 Assen startet, im gleichen Information Set. Eine Strategie σ_i für einen Spieler i ist eine Funktion, die jedem Information Set $I \in I_i$ eine Wahrscheinlichkeitsverteilung über $A(I)$ zuordnet. Diese Funktion beschreibt, wie wahrscheinlich es ist, dass Spieler i an einem Information Set eine entsprechende Aktion auswählt.
Die Menge aller möglichen Strategien für Spieler i ist definiert als Σ_i und ein Strategieprofil $\sigma \in \Sigma$ ist eine Sammlung von Strategien $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_n$. Sei $\pi^\sigma(h)$ die Wahrscheinlichkeit, dass die Historie h auftritt, wenn alle Spieler nach den Strategien $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_n$ spielen. Daraus folgt, dass der erwartete Nutzen für Spieler i, wenn alle Spieler nach σ spielen, gegeben ist durch

$$u_i(\sigma) = \sum_{z \in Z} \pi^\sigma(z) u_i(z)$$

Mit der folgenden Zerlegung erhalten wir den Beitrag jedes Spielers und des Chance-Spielers zu dieser Wahrscheinlichkeit

$$\pi^\sigma(h) = \prod_{i \in N \cup \{c\}} \pi_i^\sigma(h)$$

Die Wahrscheinlichkeit, dass Spieler i in der Strategie σ_i versucht, die Historie h zu erreichen, ist dann

$$\pi_i^\sigma(h) = \prod_{\substack{h', a \sqsubseteq h \\ P(h') = i}} \sigma_i(h', a))$$

Für die Beiträge zur Wahrscheinlichkeit eine bestimmte Historie zu erreichen für alle Spieler außer i gilt dann

$$\pi_{-i}^\sigma(h) = \prod_{j \in (N \setminus i) \cup \{c\}} \pi_j^\sigma(h)$$

Außerdem führen wir die Definition $\pi^\sigma(h, h')$ ein, die angibt, wie hoch die Wahrscheinlichkeit ist, dass eine Historie h' eintritt, wenn die Historie h gegeben ist. Die Definition von $\pi_i^\sigma(h, h')$ und $\pi_i^\sigma(h, h')$ ist analog zu $\pi^\sigma(h, h')$. Für eine detalliertere Beschreibung der Modellierungsarten möchte ich auf die Dissertation von Richard Gibson verweisen [Gib14, S. 8-10].

2.3 Counterfactual Regret Minimization

Dieser Algorithmus bringt im Moment die spielstärksten Agenten hervor und wird deshalb als Grundlage für die später betrachteten Abstraktionen verwendet. Für die Erklärung beschränke ich mich an dieser Stelle auf die grundlegende Version den sogenannten Vanilla CFR. Es hat sich allerdings gezeigt, dass die Kombination von Vanilla CFR und der Monte-Carlo-Simulation, damit nicht in jeder Iteration der komplette Spielbaum durchsucht werden muss, die Berechnungszeit im Fall von Poker deutlich reduziert [Gib14]. Alle theoretischen Garantien zur Konvergenz von Monte Carlo CFR bleiben aber identisch.

2.3.1 Regret-Minimierung

Wir gehen von einer Sequenz von Strategieprofilen $\sigma^1, \cdots, \sigma^T$ für den Spieler i aus. Der Regret für den Spieler i ist dann definiert als

$$R_i^t = \max_{\sigma_i' \in \Sigma_i} \sum_{t=1}^{T} (u_i(\sigma_i', \sigma_{-i}^t) - u_i(\sigma^t))$$

R_i^t ist der Nutzen, den Spieler i dazugewinnen könnte, wenn er der besten Strategie aus der Menge aller möglichen Strategien in allen Zeitschritten im Vergleich zu seiner tatsächlich genutzten Strategie gefolgt wäre. Ein Algorithmus minimiert den Regret von Spieler i genau dann, wenn er Strategien generiert, sodass $R_i^{T,+} \to 0$ wenn $T \to \infty$, wobei $x^+ = max\{x, 0\}$ Das Folk-Theorem beschreibt den Zusammenhang zwischen Regret-Minimierung und dem Nash-Gleichgewicht in Zero-sum-Spielen:

Es gelte $\epsilon >= 0$. In einem Nullsummenspiel mit Perfect Recall ist der Durchschnitt der Strategieprofile $\overline{\sigma}^T$ ein 2ϵ-Nash-Gleichgewicht, wenn $R_i^T / T \leq \epsilon$ für beide Spieler $i = 1, 2$. Perfect Recall bezeichnet dabei die Fähigkeit des Spielers, sich alle Informationen zu merken, die er im Verlauf des Spieles sammelt.

In einem Spiel mit Mixed-Strategieprofilen $\sigma^1, \cdots, \sigma^T$ ist das average profile für alle $i \in N, a \in A_i$ definiert als $\overline{\sigma}_i^T = \sum_{t=1}^{T} \sigma_i^t(a) \div T$. Für Spiele in der extensiven Form und verhaltensbezogenen Strategieprofilen $\sigma^1, \cdots, \sigma^T$ ist das Durchschnittsprofil für alle $i \in N, I \in \mathcal{I}_i, a \in A(I)$ definiert als

$$\overline{\sigma}_i^T(I, a) = \frac{\sum_{t=1}^{T} \pi_i^{\sigma^t}(I) \sigma_i^t(I, a)}{\sum_{t=1}^{T} \pi_i^{\sigma^t}(I)}$$

2.3.2 Regret Matching

Regret Matching ist eine einfache iterative Prozedur, die bei Spielen in der Normalform den Regret minimiert. Das initiale Profil σ^1 wird zufällig ausgewählt. Für jede Aktion $a \in A_i$ wird der akkumulierte Regret $R_i^T(a) = \sum_{t=1}^{T} (u_i(a, \sigma_{-i}^t) - u_i(\sigma_i^t, \sigma_{-i}^t))$ gespeichert, der misst, wie viel der Spieler verloren hat, weil er anstatt a zu wählen der Strategie σ_i^t gefolgt ist. Die Strategien der Folgeiterationen werden nach folgender Formel bestimmt

$$\sigma_i^{T+1}(a) = \frac{R_i^{T,+}(a)}{\sum_{b \in A_i} R_i^{T,+}(b)}$$

wobei die Aktionen zufällig ausgewählt werden, wenn der Nenner 0 wird.

2.3.3 Vanilla CFR

CFR ist ein Algorithmus, der in einem extensiven Spiel den Regret minimiert. Regret Matching wäre zwar theoretisch anwendbar, aber der Speicherplatz, der benötigt würde, um $R_i^T(a)$ für alle möglichen pure strategies $a \in A_i = S_i$ zu speichern, wäre selbst für verhältnismäßig kleine Spielbäume nicht mehr verfügbar, da $|S_i|$ exponentiell zu $|\mathscr{I}_i| * ||A(\mathscr{I}_i)|$ steigt. In CFR wird dagegen nur Speicherplatz linear zu $|\mathscr{I}_i| * |A(\mathscr{I}_i)|$ benötigt. In jeder Iteration t wird für jeden Spieler i der erwartete Nutzen für Spieler i an jedem Information Set $I \in \mathscr{I}_i$ unter dem aktuell verwendeten Strategieprofil σ^t berechnet, unter der Annahme, dass der Spieler i spielt, um das Information Set I zu erreichen. Diese Annahme nennt man den counterfactual value für Spieler i

$$u_i(I,\sigma) = \sum_{z \in Z_I} u_i(z)\pi^{\sigma}_{-i}(z[I])\pi^{\sigma}(z[I],z)$$

wobei Z_I die Menge aller Terminalzuständen ist, die in ihrer Historie das Information Set I passieren und $z[I]$ ist die Historie, die zum Terminalzustand z führt und die in ihrer Historie I enthält. Für jede Aktion $a \in A(I)$ bestimmen diese Werte den counterfactual Regret für die Iteration t, $r_i^t(I,a) = u_i(I,\sigma_{(I \Rightarrow a)}) - u_i(I,\sigma^t)$, wobei $\sigma^t_{(I \Rightarrow a)}$ dem Profil σ entspricht, außer, dass am Information Set I immer die Aktion a ausgewählt wird. Der Regret $r_i^t(I,a)$ misst, wie viel mehr Gewinn der Spieler i unter der Annahme, dass er spielt, um Information Set I zu erreichen, machen könnte, wenn er die Aktion a am Information Set I ausgewählt hätte, anstatt bei I der Strategie σ_i^t zu folgen. Diese Regrets werden summiert, um den kumulativen counterfactual Regret $R_i^T(I,a) = \sum_{t=1}^{t} r_i^t(I,a)$ zu erhalten, durch den das aktuelle Strategieprofil mit Hilfe von Regret Matching für das Information Set I bestimmt wird

$$\sigma_i^{T+1}(I,a) = \frac{R_i^{T,+}(I,a)}{\sum_{b \in A(I)} R_i^{T,+}(I,b)}$$

Für den Fall, dass der Nenner 0 wird, wird an diesem Information Set die Aktion zufällig gleichverteilt ausgewählt. Während der Berechnung wird ein kumulatives Profil $s_i^T(I,a) = \sum_{t=1}^{T} \pi_i^{\sigma^t}(I)\sigma_i^t(I,a)$ gespeichert. Wenn der Algorithmus nach T Iterationen terminiert, wird die durchschnittliche Strategie ausgegeben, die sich folgendermaßen berechnet

$$\overline{\sigma}_I^T(I,a) = \frac{s_i^T(I,a)}{\sum_{b \in A(I)} s_i^T(I,b)}$$

Für weitere Informationen zu den Garantien des CFR-Algorithmus und weiteren Varianten möchte ich auf die Doktorarbeit von Richard Gibson [Gib14] verweisen, der diesen Algorithmus sehr umfassend vorstellt.

2.4 Nash-Gleichgewicht

Vanilla CFR berechnet eine Strategie, die möglichst gut gegen eine Menge von unbekannten Gegnern spielt. Für dieses Problem ist das Nash-Gleichgewicht für ein 2-Spieler-Zero-sum-Spiel ein gängiges Lösungskonzept. Ein Nash-Gleichgewicht ist erreicht, wenn beide Spieler nach einer Strategie spielen, von der sie nicht abweichen können, ohne dass ihr erwarteter Nutzen sinkt, so lange der andere Spieler nicht von der Strategie abweicht. Sei i ein Spieler aus der Menge an Spielern des Spiels. Ein Nash-Gleichgewicht ist dann ein Strategieprofil, bei dem gilt:

$$\begin{aligned} u_1(\sigma) &\geq \max_{\sigma_1' \in \Sigma_1} u_1(\sigma_1', \sigma_2) \\ u_2(\sigma) &\geq \max_{\sigma_2' \in \Sigma_2} u_1(\sigma_2', \sigma_1) \end{aligned} \tag{2.2}$$

Für ein gegebenes $\epsilon >= 0$ ist ein Strategieprofil σ ein ϵ-Nash Gleichgewicht, wenn kein Spieler alleine von σ abweichen kann und dabei mehr als ϵ erwarteten Nutzen hinzugewinnt.

$$\begin{aligned} u_1(\sigma) + \epsilon &\geq \max_{\sigma_1' \in \Sigma_1} u_1(\sigma_1', \sigma_2) \\ u_2(\sigma) + \epsilon &\geq \max_{\sigma_2' \in \Sigma_2} u_1(\sigma_2', \sigma_1) \end{aligned} \tag{2.3}$$

2.5 Abstraktionen des Spielbaumes

Um ein Spiel in der Größe von Fixed-Limit Poker mit limitiertem Ressourceneinsatz berechnen zu können, sind Abstraktionen des Spielzustands nötig. Dieser setzt sich zusammen aus den Aktionen der Spieler in den Setzrunden und den ausgeteilten Karten zwischen den Runden. Für diese beiden Informationen gibt es verschiedene Ansätze, den Spielzustand zu abstrahieren. Außerdem gibt es die Möglichkeit, dass Informationen, die der Spieler im Laufe des Spieles gesammelt hat, wieder vergessen werden. Das wird als imperfect recall bezeichnet.

2.5.1 Card Bucketing

Vor allem im 2-Spieler-Spiel machen die möglichen Kartenkombinationen einen großen Teil des Branching-Faktors des Spielsbaumes aus. Um die Anzahl der möglichen Kartenkombinationen, die vom Algorithmus unterschieden werden zu reduzieren, wird eine Technik benutzt, die Card Bucketing genannt wird. Beim Card Bucketing werden verschiedene Hände in einem Eimer (Bucket) zusammengefasst. In jedem Bucket werden Hände zusammengefasst, die der Agent mit der gleichen Strategie spielt. Neben der Anzahl der Buckets wird der Informationsverlust auch durch die Metrik bestimmt, nach der die Karten gruppiert werden.
Eine Möglichkeit ist z. B., Hände nach ihrer Gewinnwahrscheinlichkeit zusammenzufassen. Diese Unterscheidung fasst aber möglicherweise Hände zusammen, die sich trotz ähnlicher Gewinnerwartung stark von einander unterscheiden. Betrachten wir z. B. die Hände der folgenden beiden Spieler. Der erste Spieler hält die Karten 8♢, 10♡ und auf dem Board liegen 10♣, 3♣, 4♠. Der Spieler hält mit hoher Wahrscheinlichkeit bereits die beste Hand und möchte verhindern, dass andere Spieler ihre Hände verbessern können, da die Wahrscheinlichkeit, dass seine eigene Hand sich noch weiter verbessert, gering ist. Jede hohe Karte könnte einem anderen Spieler ein besseres Paar bescheren und eine Straße oder ein Flush sind mit seinen Hole Cards nicht möglich. Der zweite Spieler hält A♠, 3♠ und auf dem Board liegen K♣, 10♠, 6♠.
Die Hand des Spielers ist in diesem Moment nicht stark, aber die Hand hat eine hohe Wahrscheinlichkeit, sich in den folgenden beiden Runden zu einem Flush zu entwickeln, also einer sehr starken Hand. Dieser Spieler hätte durch das A außerdem dann den besten möglichen Flush. Er könnte am Ende der Runde viel gewinnen, wenn es ihm gelingt, andere Spieler mit einer weniger starken Hand im Spiel zu halten und er die Hand mit weniger Verlusten verlassen könnte, sollte bis zum Ende des Spiels keine weitere Karten der gleichen Farbe gezogen werden. Die beiden Spieler werden ihre Hand also trotz gleicher Gewinnwahrscheinlichkeit sehr unterschiedlich spielen und sie mit der gleichen Strategie zu spielen würde den erwarteten Gewinn beider Hände reduzieren, wie im Beitrag [GSS07] gezeigt wird. Mögliche Metriken für die Einteilung in Buckets sind z. B. Effective Hand Strength, Effective Hand Strength Squared oder Opponent Cluster Hand Strength [Joh+13].

2.5.2 Imperfect Recall

Die Annahme, dass Spieler alle Informationen in Erinnerung behalten, die sie im Verlauf des Spieles gesammelt haben, nennt man perfect recall. Eine Relaxation dieser Annahme ist, dass der Spieler Infor-

mationen, die er im Lauf des Spieles gesammelt hat, später wieder vergisst. Das wird imperfect recall genannt. Im Formalismus eines Spieles in der extensiven Form ist das gleichbedeutend mit der Zusammenfassung von verschiedenen Information Sets zu einem gemeinsamen. Durch die geringere Anzahl an Information Sets sinkt der Branching Faktor des Spielbaumes. Dies führt jedoch dazu, dass ein Kindknoten mehr als einen Elternknoten besitzt.
Es gibt jedoch keine Garantie, dass ein Nash-Gleichgewicht für ein abstraktes Spiel mit imperfect recall durch eine Verhaltensstrategie dargestellt werden kann. Der CFR ist aber auch auf dem abstrakten Spiel wohldefiniert und kann zur Berechnung einer Strategie genutzt werden [Joh+13]. Es konnte außerdem gezeigt werden, dass eine Strategie durch die Nutzung von imperfect recall verbessert werden kann, wenn dadurch z. B. feinere Abstraktionen beim Card Bucketing möglich werden [Wau+10].

2.6 Komplexität von 2-Spieler-Fixed-Limit-Poker

Da, wie oben beschrieben, der Speicherplatzbedarf und die benötigte Rechenzeit von Algorithmen wie dem CFR in linearem Verhältnis zu der Anzahl an Information Sets in unserem Spielbaum steigt, ist es für die Berechnung der Komplexität unseres Problems entscheidend, die genaue Anzahl der Information Sets für das Spiel zu kennen. In Fixed-Limit Poker können die Spieler ihre Information Sets anhand von 2 Merkmalen unterscheiden. Die Karten, die sie auf der Hand halten (Hole und Community Cards) und die Einsätze, die die Spieler im Verlauf des Spieles gemacht haben.
Die möglichen Kartenkombinationen der einzelnen Runden lassen sich mit Hilfe kombinatorischer Formeln leicht bestimmen. Hierbei unterscheiden wir zwischen den Möglichkeiten, die es für die Reihenfolgen gibt, in der die Karten an alle Spieler ausgeteilt werden können und zwischen den Möglichkeiten, die ein einzelner Spieler unterscheiden kann, wenn er die Karten seines Gegenspielers nicht kennt. Während für die Anzahl unterscheidbarer Game States die Karten, die an alle Spieler ausgegeben werden, berücksichtigt werden, reduziert sich die Anzahl der Information Sets durch die unbekannten Karten, wenn wir das Spiel aus der Sicht eines einzelnen Spielers betrachten. Man kann die Anzahl möglicher Kartenkombinationen für einen Spieler weiter reduzieren, wenn man Kartenisomorphismen als eine nicht unterscheidbare Kartenkombination betrachtet.
Kartenisomorphismen kommen dadurch zustande, dass die Farben beim Poker keine Hirarchie haben, d. h., ein paar Asse oder ein Flush haben immer den gleichen Wert, egal aus welchen Farben die Hand besteht. Wir können Hände also unabhängig von ihren Farben gleich betrachten mit der Ausnahme, dass wir unterscheiden müssen, welche Karten die gleiche Farbe haben. So wäre z. B. eine Hand mit Karo Ass, Karo König äquivalent zu einer Hand mit Pik Ass , Pik König, aber eine Hand mit Karo Ass, Karo König, Pik 10 wäre nicht äquivalent zu einer Hand Karo Ass, Pik König, Pik 10. Wir hätten zwar die gleiche Anzahl von Karten mit einer gleichen Farbe, da aber bei einem Flush die Höhe der höchsten Karte bei einem Unentschieden mitberücksichtigt wird, um einen Gewinner zu bestimmen, hat das erste Blatt eine höhere Gewinnwahrscheinlichkeit als das zweite. Im Folgenden berechnen wir als Beispiel die Anzahl unterscheidbarer River-Hände für einen Spieler ohne die Zusammenfassung von Isomorphismen.

$$\binom{52}{2} * \binom{50}{3} * \binom{47}{1} * \binom{46}{1} = 56.189.515.200$$

Total Two-Player bezeichnet die Anzahl möglicher Kartenkombinationen beider Spieler, Total One-Player die Anzahl unterscheidbarer Kartenkombinationen für einen einzelnen Spieler und Canonical One-Player bezeichnet die Anzahl von unterscheidbaren Kartenkombinationen für einen Spieler, wobei Kartenisomorphismen zusammengefasst werden. Wie diese isomorphen Hände berechnet werden, ist in [GSS07] beschrieben. Total Two-Player dient dabei lediglich der Bestimmung aller möglichen Game States.
Die Spieler können durch fehlende Informationen über die Hole Cards der gegnerischen Spieler weniger Spielzustände unterscheiden, weshalb die Anzahl von Information Sets deutlich geringer ausfällt. Die Anzahl von Information Sets für Canonical One-Player ist dann die kleinste mögliche Größe eines abstrakten Spieles ohne Informationsverlust.

Tabelle 2.1.: Mögliche Kartenkombinationen von Hole und Public Cards in Fixed Limit Poker

Round	Total Two-Player	Total One-Player	Canonical One-Player
Preflop	1.624.350	1.326	169
Flop	28.094.757.600	25.989.600	1.286.792
Turn	1.264.264.092.000	1.221.511.200	55.190.538
River	55.627.620.048.000	56.189.515.200	2.428.287.420

Die Anzahl aller möglichen Kartenkombinationen in den jeweiligen Setzrunden sind für alle 3 Varianten in Tabelle 2.1 zusammengefasst.

Als nächstes müssen wir die Anzahl der möglichen Setzreihenfolgen bestimmen. Wir gehen hierzu davon aus, dass alle Spieler einen unbegrenzten Vorrat an Einsätzen zu Verfügung haben. D. h., die Anzahl der möglichen Aktionen wird nie durch den Vorrat an Einsätzen beschränkt. Außerdem gehen wir von einer Variante von Fixed-Limit Poker aus, bei denen die beiden Spieler je einen Small Blind, bzw. einen Big Blind vor dem Preflop, also bevor sie ihre Karten gesehen haben, bringen müssen. Die Preflop-Setzrunde ist deshalb kürzer, weil Small und Big Blind schon als Einsätze zählen und die Anzahl der möglichen Erhöhungen im Preflop um eine Erhöhung reduzieren.
Die Anzahl möglicher Setzrunden ist im 2-Spieler-Fall überschaubar, sodass man sie von Hand aufzählen kann. Außerdem werden die Knoten des Setzbaums in 4 verschiedene Knotentypen unterteilt. Sequences sind Knoten einer nicht terminalen Historie der Setzrunde. An jeder Sequence Node muss der Spieler also noch eine Aktion wählen. Eine sequence action ist eine Entscheidung für die entsprechende Aktion des Spielers an einer bestimmten Sequence Node. Auf die Sequence Node für die leere Historie kann der Spieler beispielsweise die Aktionen call, raise und fold wählen. Damit gibt es 3 mögliche sequence actions, die an dieser Sequence Node gewählt werden können.
Continuing Nodes sind Terminalhistorien einer Setzrunde, bei denen beide Spieler die gleichen Einsätze gebracht haben. Das Spiel geht hier also mit einem Unentschieden in die nächste Setzrunde.
Terminal Nodes hingegen sind Terminalhistorien, bei denen einer der Spieler das Spiel verlassen hat und der Sieger damit schon feststeht. Die Aktion fold führt also immer zu einer terminal node, während die Aktion call immer zu einer Continuing Node führt.
Da der River die letzte Setzrunde markiert, ist eine Continuing Node am River auch eine Terminal Node des Spiels, weshalb die Anzahl an Terminal Nodes am River deutlich höher ist. Bei der Aufzählung der möglichen Aktionen wurden strikt dominierte Aktionen ausgelassen. Streng genommen ist es dem Spieler erlaubt, das Spiel jederzeit zu verlassen, auch wenn er das Spiel ohne weiteren Einsatz mit der Aktion Check oder Call weiterspielen könnte. Diese Aktion führt jedoch nie zu einem höheren erwarteten Nutzen als die Aktionen Check oder Call und wurde deshalb nicht berücksichtigt. Alle Knoten der einzelnen Setzrunden mit der jeweiligen Anzahl der Knoten und den ihnen zugehörigen Historien sind in der Tabelle 2.2 aufgeführt.

Aus der Kombination von möglichen Kartenkombinationen und der Anzahl aller möglichen Setzreihenfolgen können wir jetzt die Anzahl der möglichen Information Sets, Infoset-Actions und Game States bestimmen. Die Sequence Nodes in jeder Setzrunde sind Knoten, an denen ich als Spieler eine Entscheidung treffen muss. Für jeden dieser Knoten gibt es ein Information Set für jede mögliche Hand, die der Spieler unterscheiden kann. Aus der Anzahl von Sequence Nodes und möglicher Kartenkombinationen kann man also die Anzahl von Information Sets bestimmen. Wichtig für die Berechnung des Speicherplatzes ist außerdem die Anzahl aller möglichen Aktionen innerhalb einer Setzrunde.
In einer Strategie wird für jede mögliche Aktion am Information Set eine Wahrscheinlichkeit, mit der diese gewählt wird, und ein Regret benötigt, um die Strategie für die nächste Runde Berechnen zu können. Diese Berechnen sich analog zu den Information Sets. Nutzt man für die Berechnung anstatt Total-One-Player, die Anzahl möglicher Kombinationen für Total-Two-Player erhält man die Anzahl der Game States.
Um die endgültige Anzahl der Knoten zu bestimmen, muss noch berücksichtigt werden, wie viele ver-

Tabelle 2.2.: Knoten der möglichen Setzrunden in 2-Spieler Fixed Limit Poker [Joh13, S. 5]

Round	Sequences	Actions	Continuing	Terminal
Preflop	8: -, c, cr, crr, crrr, r, rr, rrr	21: -f, -c, -r, c-c, c-r, cr-f, cr-c, cr-r, crr-f, crr-c, crr-r, crrr-f, crrr-c, r-f, r-c, r-r, rr-f, rr-c, rr-r, rrr-f, rrr-c	7: cc, crc, crrc, crrrc, rc, rrc, rrrc	7: f, rf, rrf, rrrf, crf, crrf, crrrf
Flop, Turn	10: -, c, cr, crr, crrr, crrrr, r, rr, rrr, rrrr	26: -c, -r, c-c, c-r, cr-f, cr-c, cr-r, crr-f, crr-c, crr-r, crrr-f, crrr-c, crrr-r, crrrr-f, crrrr-c, r-f, r-c, r-r, rr-f, rr-c, rr-r, rrr-f, rrr-c, rrr-r, rrrr-f, rrrr-c	9: cc, crc, crrc, crrrc, crrrrc, rc, rrc, rrrc, rrrrc	8: rf, rrf, rrrf, rrrrf, crf, crrf, crrrf, crrrrf
River	10: -, c, cr, crr, crrr, crrrr, r, rr, rrr, rrrr	26: -c, -r, c-c, c-r, cr-f, cr-c, cr-r, crr-f, crr-c, crr-r, crrr-f, crrr-c, crrr-r, crrrr-f, crrrr-c, r-f, r-c, r-r, rr-f, rr-c, rr-r, rrr-f, rrr-c, rrr-r, rrrr-f, rrrr-c	9: cc, crc, crrc, crrrc, crrrrc, rc, rrc, rrrc, rrrrc	17: cc, rc, rf, rrc, rrf, rrrc, rrrf, rrrrc, rrrrf, crc, crf, crrc, crrf, crrrc, crrrf, crrrrc, crrrrf

schiedene vorhergehende Setzrunden es gibt, bei denen das Spiel fortgesetzt wird. Da es z.B. in der Preflop-Runde 7 mögliche Continuing Nodes gibt, bei denen das Spiel fortgesetzt wird, gibt es also 7 identische Flop-Runden, nämlich eine für jede mögliche Continuing Node der Preflop-Runde. Die gesamte Anzahl aller Information Sets des 2-Spieler-Spieles für Total-One-Player berechnet sich dann wie folgt:

$$\begin{aligned} |I| &= \binom{52}{2} \times 8 \\ &+ \binom{52}{2} \times \binom{50}{3} \times 7 \times 10 \\ &+ \binom{52}{2} \times \binom{50}{3} \times \binom{47}{1} \times 7 + 9 \times 10 \\ &+ \binom{52}{2} \times \binom{50}{3} \times \binom{47}{1} \times \binom{46}{1} \times 7 \times 9 \times 9 \times 10 \\ &= 319.365.922.522.608 \end{aligned} \tag{2.4}$$

Die Gesamtzahl aller Information Sets ergibt sich also aus der Summe der Information Sets der einzelnen Setzrunden. Die Information Sets der Preflop-Runde ergibt sich aus der Anzahl aller Sequence Nodes multipliziert mit der Anzahl aller unterscheidbaren Kartenkombinationen in der Preflop-Runde. Die Anzahl der Information Sets in den späteren Runden berechnet sich, indem die Anzahl aller möglichen vorhergehenden Runden mit der Anzahl der Sequence Nodes und der unterscheidbaren Kartenkombinationen der jeweiligen Runde multipliziert wird. Auf die gleiche Art lassen sich die Anzahl möglicher Infoset-Actions und die Anzahl der Game States berechnen. Die Ergebnisse der übrigen Berechnungen sind in der Tabelle 2.3 zusammengefasst.

Tabelle 2.3.: Spielgrößen von 2-Spieler Fixed Limit Poker [Joh13, S. 6]

Betting Sequences	Round	Sequences	Sequence-Actions	Continuing	Terminal
	Preflop	8	21	7	7
	Flop	70	182	63	56
	Turn	630	1638	567	504
	River	5670	14742	0	9639
	Total	6378	16583	637	10206
One-Player Canonical	Round	Infosets	Infoset-Actions	Continuing	Terminal
	Preflop	1352	3549	1183	1183
	Flop	$9,008 * 10^7$	$2,342 * 10^8$	$8,107 * 10^7$	$7,206 * 10^7$
	Turn	$3,477 * 10^{10}$	$9,040 * 10^{10}$	$3,129 * 10^{10}$	$2,781 * 10^{10}$
	River	$1,377 * 10^{13}$	$3,580 * 10^{13}$	0	$2,341 * 10^{13}$
	Total	$1,380 * 10^{13}$	$3,589 * 10^{13}$	$3,137 * 10^{10}$	$2,343 * 10^{13}$
One-Player	Round	Infosets	Infoset-Actions	Continuing	Terminal
	Preflop	10608	27846	9282	9282
	Flop	$1,819 * 10^9$	$4,730 * 10^9$	$1,637 * 10^9$	$1,455 * 10^9$
	Turn	$7,696 * 10^{11}$	$2,001 * 10^{12}$	$6,926 * 10^{11}$	$6,156 * 10^{11}$
	River	$3,186 * 10^{14}$	$8,283 * 10^{14}$	0	$5,416 * 10^{14}$
	Total	$3,194 * 10^{14}$	$8,304 * 10^{14}$	$6,942 * 10^{11}$	$5,422 * 10^{14}$
Two-Player	Round	States	State-Actions	Continuing	Terminal
	Preflop	$1,299 * 10^7$	$3,411 * 10^7$	$1,137 * 10^7$	$1,137 * 10^7$
	Flop	$1,967 * 10^{12}$	$5,113 * 10^{12}$	$1,770 * 10^{12}$	$1,573 * 10^{12}$
	Turn	$7,965 * 10^{14}$	$2,071 * 10^{15}$	$7,168 * 10^{14}$	$6,372 * 10^{14}$
	River	$3,154 * 10^{17}$	$8,201 * 10^{17}$	0	$5,362 * 10^{17}$
	Total	$3,162 * 10^{17}$	$8,221 * 10^{17}$	$7,186 * 10^{14}$	$5,368 * 10^{17}$

Tabelle 2.4.: Speicherplatzbedarf von 2-Spieler Fixed Limit Poker

Kartenkombinationen	RAM [GB]
One Player	13.285.622
Canonical One Player	574.215

Aus der Anzahl an Infoset-Actions lässt sich der Speicherplatzbedarf des CFR-Algorithmus berechnen. Für jede infoset-action wird auf der Festplatte ein Byte Speicherplatz für die Wahrscheinlichkeit benötigt, mit der am entsprechenden informationset die Aktion ausgewählt wird. Der Algorithmus benötigt für die Berechnung der Strategie zwei 8-Byte Double-Variablen je infoset-action. Der benötigte Speicherplatz für die verschiedenen Varianten ist in der Tabelle 2.4 aufgestellt. Es würden ca. 574 GB Arbeitsspeicher für die Berechnung der Strategie benötigt, wenn als einzige Abstraktion die Kartenisomorphismen zusammengefasst werden.

Weitere Informationen dazu finden sich im Paper von Michael Johanson [Joh13].

2.7 CFR in Multiplayer-Fixed-Limit-Poker

Die Agenten der University of Alberta setzen schon seit einigen Jahren erfolgreich CFR-Algorithmen für die Berechnung von 3-Spieler-Fixed-Limit-Strategien ein. Auch wenn es keine Garantien für die Existenz eines Nash-Gleichgewichtes für Multiplayer-Fixed-Limit gibt, konnten sie nachweisen, dass der Algorithmus im Mehrspielerfall keine Strategien produziert, die von anderen Strategien dominiert werden. Zwei Agenten mit verschiedenen Kartenabstraktionen wurden mit Hilfe von CFR berechnet. Eine Variante mit Imperfect Recall und 16 Buckets in jeder Runde und eine Variante mit Perfect Recall und 2 Buckets in jeder Runde (2,4,8,16 Buckets in den Runden Preflop bis River). Um den Spielbaum zusätzlich zu verkleinern wurde die River-Runde anstatt maximal 4 Bets, auf maximal 3 Bets begrenzt [Ris, S. 28]. Diese Abstraktion reduziert die Größe des Spielbaumes deutlich, verändert die Strategie der Agenten aber kaum. Diese Agenten wurden dann im Spiel mit anderen 3-Spieler-Agenten verglichen [Ris, S. 32-35]. Dabei zeigten sich leichte Vorteile des Imperfect-Recall-Agenten gegenüber der Perfect-Recall-Variante und sie waren selbst mit diesen sehr groben Abstraktionen bereits in der Lage, PsyOpti zu schlagen. Außerdem wurde eine Technik namens Strategy Stitching verwendet, mit der unterschiedliche Strategien für Teilbäume des Spielbaumes zu einer gemeinsamen Strategie verknüpft werden können. Für das Strategy Stitching wurde auch der Ansatz verfolgt, Strategien ausgehend von Spielsituationen, in denen ein Spieler ausgestiegen ist, getrennt zu berechnen. Diese Strategien für Teilbäume werden Heads-up-Experts genannt.

Die bisher spielstärksten Varianten, die auch bei der Annual Computer Poker Competition teilgenommen haben, verwenden eine Strategie, die mit Hilfe des PureCFR Algorithmus berechnet wurde, der für die Berechnung von Strategien Pure anstatt Mixed Strategies verwendet, um teure Floating-Point-Berechnungen durch Integer-Operationen zu ersetzen [Coma]. Darüber hinaus wird Strategy Stitching verwendet. In diesen Agenten werden innerhalb der Stratgie sehr viel mehr Buckets für häufig auftretende Spielsituationen verwendet. Weitere Abstraktionen werden für diese Agenten nicht beschrieben. Das Hauptaugenmerk bei den verwendeten Abstraktionen liegt also weiterhin auf den Kartenabstraktionen. Die Heads-up-Experten werden in dieser Arbeit ebenfalls noch genauer evaluiert. Eine detaillierte Komplexitätsanalyse, wie sie für den 2-Spieler-Fall in Abschnitt 2.6 beschrieben wird, gibt es für 3-Spieler-Fixed-Limit noch nicht.

3 Komplexität von 3-Spieler-Fixed-Limit-Poker

Um den Nutzen der Abstraktionen, die ich hier untersuchen möchte, bewerten zu können ist es nötig, eine detallierte Komplexitätsanalyse des 3-Spieler-Spielbaumes durchzuführen.

3.1 Berechnungsverfahren

Im Gegensatz zur 2-Spieler-Variante ist die Anzahl möglicher Setzrunden nur mit großem Aufwand händisch zu bestimmen. Eine weitere Besonderheit ergibt sich aus der Berechnung späterer Setzrunden mit Hilfe der Continuing Nodes aus vorhergehenden Runden. Im 2-Spieler-Spiel gibt es nur 2 Knoten, die unterschieden werden müssen. Continuing Nodes und Terminal Nodes. Im 3-Spieler-Fall würde diese Unterscheidung zu einem Rechenfehler führen.
Nehmen wir an, ein Spieler verlässt das Spiel in der Preflop-Runde. Die anderen beiden Spieler bleiben weiter im Spiel und erreichen die Flop-Runde. Die beiden verbleibenden Spieler bieten jetzt am Flop, wie es in einem 2-Spieler-Spiel der Fall wäre, da der dritte Spieler in allen verbleibenden Runden keine Aktionen mehr tätigen kann. Das bedeutet, der Setzbaum für die folgenden Runden ist äquivalent zum 2-Spieler-Setzbaum und damit um ein Vielfaches kleiner als der eines Spieles mit 3 aktiven Spielern. Um diese Ungenauigkeit zu beseitigen, unterscheiden wir die Knoten innerhalb einer Setzrunde zusätzlich nach der Zahl der Spieler, die noch aktiv am Spiel teilnehmen. Continuation Nodes mit zwei aktiven Spielern, werden als 2-Spieler-Spiele zu Ende gespielt und Spiele mit drei aktiven Spielern mit dem kompletten 3-Spieler-Baum fortgesetzt.
Im Folgenden bezeichnen wir mit 3PG und 2PG, mit wie vielen aktiven Spielern die Setzrunde begonnen hat. 3P und 2P bezeichnen die Anzahl von aktiven Spielern an den jeweiligen Knoten. Die Spielrunden kürzen wir mit PF für Preflop, F für Flop, T für Turn und R für River ab. Die Art der Knoten bezeichnen wir mit Cons für Continuing Nodes, Seq für Sequence Nodes, SeqAct für sequence actions und Term für terminating nodes. 3PG-2P-T-Cons wären z. B. die Anzahl von Continuing Nodes in der Turn-Runde, die mit 3 aktiven Spielern beginnt und an der noch 2 Spieler aktiv am Spiel teilnehmen. Die Anzahl aller möglichen Sequence Nodes am Flop berechnen wir folgendermaßen:

$$\begin{aligned} \text{2P-Flop-Sequences} &= \text{3PG-2P-PF-Cons} * \text{2PG-2P-F-Seq} + \text{3PG-3P-PF-Cons} * \text{3PG-2P-F-Seq} \\ \text{3P-Flop-Sequences} &= \text{3PG-3P-PF-Cons} * \text{3PG-3P-F-Seq} \\ \text{Total Flop Sequences} &= \text{2P-Flop-Sequences} + \text{3P-Flop-Sequences} \end{aligned} \tag{3.1}$$

Auch alle anderen Arten von Knoten für die Setzrunden berechnen wir auf diese Art. Dazu berechnen wir für jede Runde den Knotentyp für alle möglichen Zahlen von aktiven Spielern. Bei späteren Runden und Knoten, bei denen nur noch 2 Spieler aktiv spielen, müssen wir alle möglichen Runden betrachten, in denen der dritte Spieler das Spiel verlassen haben könnte. Da die Aufzählung der Knoten und die Unterscheidung in die unterschiedlichen Knotenarten mit steigender Spielerzahl unhandlich wird, habe ich ein Programm geschrieben, das einen Setzbaum für eine bestimmte Setzrunde für Fixed-Limit Poker mit variabler Spielerzahl aufbauen kann.

3.2 Programm zur Aufzählung von Fixed-Limit-Setzrunden

Für die Aufzählung der verschiedenen Knotentypen einer Setzrunde müssen wir nicht den gesamten Spielbaum aufbauen. Es genügt, wenn wir alle unterschiedlichen Setzrunden einzeln darstellen können. Das Programm ermöglicht die freie Definition der Ausgangssituation für die Setzrunde. Es ist möglich den Startspieler und die bereits eingesetzen Bets anzupassen. So ist es nicht nur möglich Aufzählungen

Tabelle 3.1.: Anzahl der Knotentypen des Setzbaumes von 3-Spieler Fixed Limit Poker

Round	Preflop	Flop	Turn	River
Total Sequences	82	171	171	171
2-Player Sequences	37	78	78	78
3-Player Sequences	45	93	93	93
Total Actions	199	417	417	417
2-Player Actions	89	189	189	189
3-Player Actions	110	228	228	228
Total Continuing Nodes	81	169	169	169
2-Player Continuing Nodes	59	123	123	123
3-Player Continuing Nodes	22	46	46	46
Terminal Nodes	37	78	78	247

für die vollständigen Setzrunden zu erstellen, sondern auch nur Teilbäume, ausgehend von einer bestimmten Spielsituation. Nach dem Aufbau des Baumes ist es möglich die Anzahl der unterschiedlichen Knotentypen, sowie alle Aktionen der Spieler aufzuzählen, die zum jeweiligen Knoten geführt haben.
Der Betting Tree Generator besteht aus 2 Klassen. Die Klasse GameStateNode enthält die Logik für den Aufbau und das Traversieren über den Spielbaum und die Klasse GameRules definert Spielregeln, wie die Ausgangssituation der Runde und die maximale Anzahl an Bets.
Der Source Code der Klassen befindet sich in Anhang C. Die Klassen wurden in Java geschrieben.

3.3 Ergebnisse

In den folgenden Tabellen ist die Anzahl der Knotentypen in den einzelnen Setzrunden aufgeschlüsselt. Die Ausgangssituation ist hierbei immer eine Runde, in der alle 3 Spieler noch aktiv am Spiel teilnehmen. Wenn eine der späteren Runden nur noch zwischen 2 aktiven Spielern gespielt wird, kann die entsprechende Kontenzahl aus der 2-Spieler-Komplexitätsanalyse verwendet werden. Da die Tabelle der Knoten der 3-Spieler-Runde ungefähr um den Faktor 10 größer ist, als in der 2-Spieler-Variante wird hier auf die Aufstellung aller Historien der Sequences, sequence actions, continuing und Terminal Nodes in Tabellenform verzichtet. Eine Auflistung aller Knotentypen mit den zugehörigen Spielhistorien für die 3-Spieler-Variante findet sich im Anhang A, zusammen mit den übrigen hier nicht aufgeführten Berechnungen. In der Tabelle 3.1 ist lediglich die Anzahl der entsprechenden Knoten in den einzelnen Setzrunden aufgeführt.

Die Anzahl der Sequences sind ungefähr um den Faktor 10 höher als im 2-Spieler-Fall, die Anzahl der Continuing Nodes steigen um das Vierfache. Aus der Anzahl der verschiedenen Knoten innerhalb der einzelnen Setzrunden berechnen wir jetzt die Anzahl der Gesamtknoten im Setzbaum. Als Beispiel ist in der Tabelle 3.2 das Berechnungsverfahren für die Anzahl von Sequence Nodes im gesamten Setzbaum beschrieben. Die weiteren Berechnungen funktionieren nach dem gleichen Schema. Die Gesamtzahl der Sequences und Sequence-Actions sind in den Tabellen 3.3 und 3.4 zusammengefasst. Auf die Betrachtung aller Continuing Nodes und Terminal Nodes wie im 2-Spieler-Spiel verzichte ich aufgrund der ohnehin schon großen Zahl an Tabellen in den folgenden Betrachtungen, da sie für die Berechnung der entscheidenden Größen nicht notwendig sind.
Mit der Gesamtzahl aller Knoten im Setzbaum können wir jetzt die Anzahl aller möglichen Information Sets, Infoset-Actions, continuing und Terminal Nodes bestimmen. Die Anzahl möglicher Kartenkombinationen für das 3-Spieler-Spiel ist in der Tabelle 3.5 aufgeführt. Die Kartenkombinationen für 1-Player und Canonical-1-Player sind identisch zur 2-Spieler-Variante, da der Spieler weiterhin keine der Karten seiner Gegenspieler sehen kann. Die Anzahl für 3-Player und damit die Anzahl der möglichen Spielzustände steigt dagegen beträchtlich an. Zusätzlich dazu wird ein Spieler mit dem kleinsten möglichen Perfect-

Tabelle 3.2.: Berechnung der Knoten des Setzbaumes von 3-Spieler Fixed Limit Poker

2 Active Players	
Preflop	
3PG-2P-P-Seq	37
Flop	
3PG-3P-PF-Cons * 3PG-2P-F-Seq	1.716
3PG-2P-PF-Cons * 2PG-F-Seq	590
Total	2.306
Turn	
3PG-2P-PF-Cons * 2PG-F-Cons * 2PG-T-Seq	5.310
3PG-3P-PF-Cons * 3PG-2P-F-Cons * 2PG-T-Seq	27.060
3PG-3P-PF-Cons * 3PG-3P-F-Cons * 3PG-2P-T-Seq	78.936
Total	111.306
River	
3PG-2P-PF-Cons * 2PG-F-Cons * 2PG-T-Cons * 2PG-R-Seq	47.790
3PG-3P-PF-Cons * 3PG-2P-F-Cons * 2PG-T-Cons * 2PG-R-Seq	243.540
3PG-3P-PF-Cons * 3PG-3P-F-Cons * 3PG-2P-T-Cons * 2PG-R-Seq	1.244.760
3PG-3P-PF-Cons * 3PG-3P-F-Cons * 3PG-3P-T-Cons * 3PG-2P-R-Seq	3.631.056
Total	5.167.146
3 Active Players	
Preflop	
3PG-3P-PF-Seq	45
Flop	
3PG-3P-PF-Cons * 3PG-3P-F-Seq	2.046
Turn	
3PG-3P-PF-Cons * 3PG-3P-F-Cons * 3PG-3P-T-Seq	94.116
River	
3PG-3P-PF-Cons * 3PG-3P-F-Cons * 3PG-3P-T-Cons * 3PG-3P-R-Seq	4.329.336

Tabelle 3.3.: Sequences von 3-Spieler Fixed Limit Poker

Round	Sequences	2-Player Sequences	3-Player Sequences
Preflop	82	37	45
Flop	4.352	2.306	2.046
Turn	205.422	111.306	94.116
River	9.496.482	5.167.146	4.329.336
Total	9.706.338	5.280.795	4.425.543

Tabelle 3.4.: Sequence-Actions von 3-Spieler Fixed Limit Poker

Round	Sequence-Actions	2-Player Sequence-Actions	3-Player Sequence-Actions
Preflop	199	89	110
Flop	10.708	5.692	5.016
Turn	506.166	275.430	230.736
River	23.406.018	12.792.162	10.613.856
Total	23.923.091	13.073.373	10.849.718

Recall-Card-Bucketing hinzugefügt. Der Spieler unterscheidet in jeder Runde nur ob das Blatt eine über- oder unterdurchschnittliche Gewinnerwartung hat und sortiert das Blatt in den entsprechenden Bucket ein. Dieses Bucketing wurde für die Berechnung der ersten 3-Spieler-CFR-Strategien genutzt und soll die Grenzen des Card Bucketing aufzeigen [Ris].

Zuletzt werden die Anzahl möglicher Kartenkombinationen der Spieler mit der Gesamtzahl der einzelnen Knoten in den jeweiligen Setzrunden mutlipliziert, um die Anzahl an Information Sets, Infoset-Actions, bzw. Game States zu berechnen. Die Ergebnisse sind in der Tabelle 3.7 aufgeführt.

3.4 Analyse der Ergebnisse

Die Gesamtzahl der Information Sets und damit auch Speicherplatz und Zeit, die für die Berechnung einer Strategie mit dem CFR-Algorithmus nötig wären, sind für den Spieler mit Kartenisomorphismen somit ungefähr um den Faktor 1000 größer als im 2-Spieler-Fall. Damit ist das Problem so groß, dass es nur noch mit sehr groben Kartenabstraktionen gelöst werden kann, wie man am Speicherplatzbedarf für den Bucketing-Spieler ablesen kann. Der dominante Faktor ist die Anzahl der Information Sets und Infoset-Actions in den River-Runden, neben denen die Anzahl der übrigen Information Sets vernachlässigbar ist. Die Anzahl von Information Sets für die River-Runde zu reduzieren hat also den größten Effekt auf die Gesamtgröße des Spielbaumes. Dazu ist des nötig, entweder die Anzahl der Continuing Nodes der vorhergehenden Runden oder die Anzahl der Sequence Nodes in der River-Runde zu reduzieren. Ein Beispiel hierfür findet sich in 3-Spieler-CFR-Varianten der University of Alberta, in denen die River-Runde auf maximal 3 Einsätze beschränkt wurde, um den gesamten Baum im Speicher unterbringen zu können [Ris, S. 28]. Ein möglicher vierter Einsatz an der River-Runde wurde von dem Agenten immer gecalled.

Tabelle 3.5.: Kartenkombinationen der Spielertypen für 3-Spieler Fixed Limit Poker

Round	3-Player	1-Player	Canonical-1-Player	Card Bucketing
Preflop	$1,83 * 10^{09}$	$1,33 * 10^{03}$	$1,69 * 10^{02}$	2
Flop	$2,78 * 10^{13}$	$2,60 * 10^{07}$	$1,29 * 10^{06}$	4
Turn	$1,20 * 10^{15}$	$1,22 * 10^{09}$	$5,52 * 10^{07}$	8
River	$5,02 * 10^{16}$	$5,62 * 10^{10}$	$2,43 * 10^{09}$	16

Tabelle 3.6.: Informationsets und Game States von 3-Spieler Fixed Limit Poker

Kartenkombinationen	Runde	Game States	Game State Actions
3-Player-Game	Preflop	$1,56 * 10^{14}$	$5,86 * 10^{14}$
	Flop	$1,09 * 10^{20}$	$3,95 * 10^{20}$
	Turn	$2,27 * 10^{23}$	$7,99 * 10^{23}$
	River	$3,91 * 10^{26}$	$1,40 * 10^{27}$
	Total	$3,91 * 10^{26}$	$1,40 * 10^{27}$
Kartenkombinationen	Runde	Infosets	Infoset-Actions
1 Player - 3-Player-Game	Preflop	$1,09 * 10^{05}$	$4,10 * 10^{05}$
	Flop	$1,09 * 10^{05}$	$4,09 * 10^{11}$
	Turn	$2,69 * 10^{14}$	$9,48 * 10^{14}$
	River	$5,34 * 10^{17}$	$1,91 * 10^{18}$
	Total	$5,34 * 10^{17}$	$1,91 * 10^{18}$
1 Canonical Player - 3-Player-Game	Preflop	$1,39 * 10^{04}$	$5,22 * 10^{04}$
	Flop	$5,60 * 10^{09}$	$2,02 * 10^{10}$
	Turn	$1,22 * 10^{13}$	$4,28 * 10^{13}$
	River	$2,31 * 10^{16}$	$8,26 * 10^{16}$
	Total	$2,31 * 10^{16}$	$8,27 * 10^{16}$
1 Bucketing Player - 3-Player-Game	Preflop	$1,64 * 10^{02}$	$6,18 * 10^{02}$
	Flop	$1,74 * 10^{04}$	$6,29 * 10^{04}$
	Turn	$1,76 * 10^{06}$	$6,21 * 10^{06}$
	River	$1,52 * 10^{08}$	$5,44 * 10^{08}$
	Total	$1,54 * 10^{08}$	$5,51 * 10^{08}$

Tabelle 3.7.: Speicherbedarf von 3-Spieler-Fixed-Limit-Poker

Kartenkombinationen	RAM Speicherbedarf [GB]
1 Player – 3 Player Game	30.600.143.688, 581
1 Canonical Player- 3-Player Game	1.322.446.409, 212
1 Bucketing Player- 3-Player Game	8, 809

4 Truncated Multiplayer Nodes

Ein Problem des Einsatzes von CFR für die Lösung von Mehrspielerspielen ist, dass es keine theoretischen Garantien für die Existenz einer Nash-optimalen Lösung gibt. Wir werden hier jedoch zeigen, dass diese Garantien weiterhin für Teilbäume gelten, in denen der dritte Spieler ausgestiegen ist und das Spiel nur noch mit 2 Spielern zu ende gespielt wird. Ein möglicher Abstraktionsansatz ist es deshalb, sich auf die Lösung der 2-Spieler-Teilbäume zu beschränken und die Entscheidung an Mehrspielerknoten einem anderen Agenten zu überlassen. Bei einer Kombination mehrerer Agenten spricht man von einem Expertensystem. Solche Systeme wurden in einer vorhergehenden Arbeit bereits untersucht. Expertensysteme finden in der Poker-Domäne schon länger Anwendung, unter anderem zur Kombination von Agenten, die gegen bestimmte Gegner trainiert wurden. Ein Problem dabei ist der Aufbau aller Teilbäume, an denen nur noch 2 Spieler aktiv spielen. Dazu könnte man alle möglichen Zustände als Startzustände für den Algorithmus als Start eines neues Spieles zwischen den 2 Spielern festlegen. Während diese Möglichkeit die Anzahl der Information Sets am stärksten reduzieren würde, sind die Identifizierung aller Startzustände und die Änderungen, die am Algorithmus und einem CFR-Spieler notwendig wären, sehr umfangreich. Eine Alternative hierzu wäre, unterschiedliche Kartenabstraktionen für Spielsituationen zwischen 2 und 3 Spielern einzusetzen. In einem 3-Spieler-Knoten würde der Spieler mit einer sogenannten blind abstraction spielen, d. h. er würde die Karten, die ihm eigentlich bekannt sind, nicht betrachten. Erst wenn einer der Spieler aussteigt, würden die beiden verbleibenden Spieler mehrere Buckets für die Kategorisierung der Karten nutzen. Diese Variante hat zwar mehr Information Sets, aber die Zahl der zusätzlichen Information Sets, die notwendig sind, um den Spielbaum bis zum Aussteigen eines Spielers aufzubauen, ist vernachlässigbar klein gegenüber der Anzahl an Information Sets des gesamten Spielbaumes. Für diese Variante müsste der Algorithmus nicht grundlegend verändert werden, er könnte mit gängigen Multiplayer-Implementierungen des CFR-Algorithmus ohne große Änderungen durchgeführt werden. Auch der CFR-Agent, der nach dieser Strategie spielt, müsste nicht angepasst werden.

4.1 Nash-Gleichgewicht in Non-Zero-Sum-Games

Ein Nash-Gleichgewicht ist in Non-Zero-Sum-Games nicht zwangsläufig ein geeignetes Lösungskonzept. Warum das so ist, kann man am Beispiel einer Variante des Spieles Kopf oder Zahl nachvollziehen. Das Spiel wird mit 3 Spielern gespielt. Jeder Spieler wählt entweder das Symbol Kopf oder Zahl, ohne dass die anderen Spieler seine Entscheidung kennen. Wählen alle Spieler das gleiche, behält jeder der Spieler seinen Einsatz. Wählt einer der Spieler ein anderes Symbol, als die beiden anderen erhält er ihre Einsätze. Ein mögliches Nash-Gleichgewicht für dieses Spiel ist gegeben durch $Sigma = \{\sigma_1, \sigma_2, \sigma_3\}$ wobei σ_1 mit Wahrscheinlichkeit 1 Kopf wählt und σ_2 und σ_3 mit Wahrscheinlichkeit 1 Zahl. Dieses Strategieprofil ist ein Nash-Gleichgewicht, da kein Spieler alleine von der Strategie abweichen kann und dabei seinen erwarteten Gewinn verbessert. Spieler 3 könnte aber durchaus seine Strategie anpassen um den erwarteten Nutzen von Spieler 1 zu reduzieren und den von Spieler 2 zu erhöhen, ohne selbst an Nutzen einzubüßen. Trotz des Nash-Gleichgewichtes kann Spieler 1 also nicht sicher sein, dass seine gewählte Strategie optimal ist [Gib14, S. 39].

4.2 Theoretische Garantien

Für den gesamten Spielbaum gilt das gleiche wie für die unveränderte 3-Spieler-Variante des CFR-Algorithmus. Die theoretischen Garantien aus der 2-Spieler-Variante gelten hier nicht mehr. Da der Agent aber nur Situationen entscheiden soll, die zwischen 2 Spielern stattfinden, interessieren uns nur die Teilbäume der Knoten mit 2 aktiven Spielern. Die Frage ist also, ob der Algorithmus für die Teilbäume mit

2 aktiven Spielern zu einer optimalen Lösung konvergiert. Ein Nullsummen-Spiel ist definiert als ein mit Spiel zwischen 2 Spielern für deren Nutzenfunktionen $u_1 + u_2 = 0$ gelten muss. Diese Annahme ist hier offensichtlich verletzt. In allen übrigen Knoten gibt es zwar nur noch 2 aktive Spieler, der Nutzen des siegreichen Spielers ist aber um den Einstatz des ausgestiegenen dritten Spielers höher, als der Verlust des anderen. Die Annahme wird also verletzt. Wir können das Spiel jedoch so umformulieren, dass das Aussteigen eines Spielers als der Beginn eines neuen Spieles zwischen den beiden verbleibenden Spielern angesehen wird. Der neue Spielzustand ist jetzt kein Nullsummenspiel mehr. Wir können das Spiel allerdings in ein Konstant-Summen-Spiel umformulieren, indem wir den Einsatz des ausgestiegenen Spielers auf den Nutzen der beiden verbliebenen Spieler aufaddieren. Sei der Einsatz des dritten Spielers bezeichnet mit g und u_1, u_2 die Nutzenfunktionen der beiden im Spiel verbliebenen Spieler, so wird aus dem Spiel das Konstantsummenspiel mit der Summe $g = u_1 + u_2$, wobei der Nutzen beider Spieler sowohl positiv, als auch negativ sein kann. Das Konstantsummenspiel können wir allerdings durch die Einführung eines Faktors $\frac{1}{n} * g$, wobei n die Anzahl der im Spiel verbleibenden Spieler bezeichnet, in ein Nullsummenspiel umformen. Diesen Faktor ziehen wir von der Nutzenfunktion der Spieler ab und transformieren unser Spiel damit wieder in ein Nullsummenspiel, in dem die theoretischen Garantien für die Existenz einer Nash-optimalen Lösung wieder gelten. $(u_1 - \frac{1}{2} * g) + (u_2 - \frac{1}{2} * g) = 0$ Eine Nash-optimale Lösung für dieses transformierte Spiel kann dann als approximierte Lösung für das Ausgangsspiel genutzt werden. Mit dieser Variante sollte man in der Lage sein die 2-Spieler-Teilbäume auch innerhalb des 3-Spieler-Spiels mit den Garantien der 2-Spieler-Variante lösen zu können. Jedoch ist nicht abzusehen, wie der Algorithmus auf die veränderte Utility Funktion in den 2-Spieler-Knoten reagiert und was das für die Konvergenz des Algorithmus insgesamt bedeutet.

4.3 Komplexitätsreduktion durch die Abstraktion

Um abschätzen zu können, ob die Abstraktion sinnvoll eingesetzt werden kann ist es notwendig zu berechenen, um welchen Faktor sich die Anzahl der Information Sets reduziert, wenn wir die Abstraktion auf den Spielbaum anwenden. Dazu nutzen wir die Komplexitätsanalyse aus dem vorherigen Kapitel.
Bei der Berechnung der Größe des resultierenden Spielbaumes genügt es, Änderungen bei der Berechnung der Information Sets vorzunehmen. Statt die Gesamtanzahl der Knoten einer Setzrunde mit der Anzahl der Kartenkombinationen zu multiplizieren, unterscheiden wir bei der Berechnung jetzt zwischen 2- und 3-Spieler-Knoten. Sei Seq_i^r die Anzahl der Sequenzen mit i aktiven Spielern in der Runde r und k^r die Anzahl möglicher Kartenkombinationen in Runde r. Die Anzahl der Informationsets am River berechnet sich dann folgendermaßen:

$$|\mathcal{I}| = Seq_2^4 * k^4 + Seq_3^4$$

Die Ergebnisse des 3-Spieler-Fixed-Limit-Poker-Spieles mit Truncated Multiplayer Nodes sind in den Tabellen 4.1 und 4.2 zusammengefasst.

4.4 Interpretation der Ergebnisse

Wie gut zu erkennen ist, reduziert sich die Anzahl der gesamten Information Sets nur ca. um den Faktor 2, wegen der großen Anzahl von 2-Spieler-Knoten im Verhältnis zur Gesamtgröße des Spielbaumes. Außerdem verliert man viele Informationen über das Spiel. Desweiteren wurde, obwohl CFR nicht unbedingt zu einem Nash-Gleichgewicht konvergiert, bereits experimentell gezeigt, dass CFR gute Strategien in 3-Spieler-Fixed-Limit erreicht. Einen Ansatz warum das so sein könnte, liefert Richard Gibson in seiner Arbeit [Gib14]. Bei der Komplexitätsanalyse von TMN ist allerdings die Idee für eine Abstraktion entstanden, die die Komplexität deutlich stärker reduziert und die im folgenden Kapitel vorgestellt wird.

Tabelle 4.1.: Information Sets und Game States von 3-Spieler-Fixed-Limit-Poker mit TMN

Kartenkombinationen	Runde	Game States	Game State Actions
3-Players	Preflop	$7,02 * 10^{13}$	$1,69 * 10^{14}$
	Flop	$5,79 * 10^{19}$	$1,43 * 10^{20}$
	Turn	$1,30 * 10^{23}$	$3,24 * 10^{23}$
	River	$2,13 * 10^{26}$	$5,27 * 10^{26}$
	Total	$2,13 * 10^{26}$	$5,27 * 10^{26}$
Kartenkombinationen	Runde	Infosets	Infoset-Actions
1-Player	Preflop	$4,91 * 10^{04}$	$1,18 * 10^{05}$
	Flop	$5,99 * 10^{10}$	$1,48 * 10^{11}$
	Turn	$1,55 * 10^{14}$	$3,85 * 10^{14}$
	River	$2,90 * 10^{17}$	$7,19 * 10^{17}$
	Total	$2,90 * 10^{17}$	$7,19 * 10^{17}$
Canonical-1-Player	Preflop	$6,30 * 10^{03}$	$1,52 * 10^{04}$
	Flop	$2,97 * 10^{09}$	$7,32 * 10^{09}$
	Turn	$6,98 * 10^{12}$	$1,74 * 10^{13}$
	River	$1,25 * 10^{16}$	$3,11 * 10^{16}$
	Total	$1,26 * 10^{16}$	$3,11 * 10^{16}$
Bucketing-1-Player	Preflop	$1,19 * 10^{02}$	$2,88 * 10^{02}$
	Flop	$1,13 * 10^{04}$	$2,78 * 10^{04}$
	Turn	$1,11 * 10^{06}$	$2,75 * 10^{06}$
	River	$8,70 * 10^{07}$	$2,15 * 10^{08}$
	Total	$8,81 * 10^{07}$	$2,18 * 10^{08}$

Tabelle 4.2.: Speicherbedarf 3-Spieler Fixed Limit Poker mit TMN

Kartenkombinationen	RAM Speicherbedarf [GB]
1 Player – 3 Player Game	11.506.722.888, 279900
1 Canonical Player- 3-Player Game	497.286.924, 552352
1 Bucketing Player- 3-Player Game	3, 489063

5 Imperfect Betting Recall

Bei der Berechnung der Anzahl von Information Sets des 3-Spieler-Spiels wurde zuvor bereits gezeigt, dass der dominante Faktor für die Größe des Problems die Anzahl von Information Sets am River ist. Dieser Faktor berechnet sich aus der Anzahl an Sequence Nodes am River und der Multiplikation von allen Continuing Nodes der vorangegangenen Runden. Die Anzahl der Continuing Nodes geht dabei also mehrfach in die Berechnung des Faktors ein. Wenn es uns gelingt, die Anzahl der Continuing Nodes der einzelnen Runden zu reduzieren, könnten wir damit auch den Spielbaum deutlich verkleinern. Dafür möchte ich das Konzept Imperfect Recall, das schon bei Kartenabstraktionen erfolgreich angewendet wurde, auf die Spielzustände am Ende der einzelnen Setzrunden übertragen, um die Anzahl von Continuing Nodes zu beschränken. Bei Informationen, die dem Spieler zur Verfügung stehen, unterscheidet man zwischen Perfect und Imperfect Recall. Beim Perfect Recall merkt sich der Spieler alle Informationen, die im Verlauf des Spiels gesammelt wurden. Beim Imperfect Recall vergisst der Spieler Teile dieser Informationen über das Spiel, um den Zustandsraum zu verkleinern. Angewandt auf die Kartenabstraktionen bedeutet das, der Spieler unterscheidet bei seinen Händen nicht mehr wie stark seine Hand in der vorangegangenen Runde war. Er vergisst, in welchem Bucket die Karten in der vorangegangenen Setzrunde einsortiert wurden. Während bei Perfect Recall in jeder Setzrunde die Anzahl der Buckets deshalb exponentiell steigt, kann beim Imperfect Recall die Anzahl der Buckets für jede einzelne Runde frei gewählt werden. Insbesondere kann eine feinere Abstraktion für die verhältnismäßig wenigen Information Sets in den frühen Setzrunden gewählt werden ohne, dass die Anzahl der Buckets in späteren Runden extrem hoch wird. Um die Anzahl an Continuing Nodes zu reduzieren, fasse ich äquivalente Spielzustände am Ende einer Setzrunde zu einem gemeinsamen Knoten zusammen. Die Zustände werden nach der Anzahl der aktiven Spieler und der Anzahl der Erhöhungen im Pot unterschieden. Dabei vergisst der Spieler, in welcher Reihenfolge die einzelnen Spieler in der vorangegangenen Runde erhöht haben.
Eine besondere Eigenschaft der Bucketing Techniken ist, dass der Abstraktionsgrad angepasst werden kann, was bei TMN z.B. nicht der Fall ist. Deshalb soll hier außerdem gezeigt werden, dass Imperfect Betting Recall ebenfalls diese Eigenschaft besitzt. Die Information, welcher Spieler in der vorherigen Runde als letzter erhöht hat, würde in dieser Abstraktion verloren gehen. Diese Information könnte aber z. B. einen Einfluss darauf haben, ob ein Bluff in der folgenden Runde erfolgreich ist, weil der Spieler in der vorangegangenen Runde eine starke Hand signalisiert hat. Diese Information sollte deshalb im Spielzustand erhalten bleiben, obwohl der Spielbaum dadurch nicht ganz so stark verkleinert wird. Es wäre aber durchaus denkbar, diese Information nicht zu berücksichtigen, bzw. noch weitere Informationen über die Setzrunde beizubehalten, um den Abstraktionsgrad variieren zu können.
Nehmen wir als Beispiel für die Funktionsweise der Abstraktion 2 mögliche Spielverläufe zwischen 3 Spielern an. In beiden Spielen hat noch keiner der Spieler in der Runde einen Einsatz gebracht. Im ersten Spielverlauf raised erst Spieler 2 und dann Spieler 3 ein weiteres Mal, worauf alle anderen Spieler callen. Im zweiten Spielverlauf raised der erste Spieler, der zweite Spieler geht die Erhöhung mit, der dritte Spieler raised ein zweites Mal und die anderen beiden Spieler callen. Am Ende der Runde haben in beiden Spielen die Spieler jeweils 2 Bets eingesetzt, alle 3 Spieler sind noch im Spiel. Diese beiden Zustände wären bei Imperfect Betting Recall äquivalent und würde zur gleichen Continuing Node führen. Diese Art der Abstraktion hat außerdem noch einen interessanten Nebeneffekt. Gerade die Anzahl der möglichen Continuing Nodes, bei der alle der Spieler noch am Spiel teilnehmen, reduziert sich am stärksten, da kein Spieler im Verlauf der Setzrunde das Spiel verlassen hat. Imperfect Betting Recall hat die gleichen Einschränkungen wie andere Imperfect-Recall-Abstraktionen. Es gibt keine Garantie mehr für die Existenz einer nash-optimalen Verhaltensstrategie [Joh16, S. 77], aber wie bereits in den Grundlagen erwähnt, bleibt CFR auf einem abstrakten Spiel mit Imperfect Recall weiter wohldefiniert und die Berechnung einer Strategie ist möglich.

5.1 Berechnungsverfahren

Für die Berechnung der reduzierten Continuing Nodes betrachten wir alleine die Einsätze der Spieler nach dem Ende einer Setzrunde, die noch im Spiel befindlichen Spieler und den Spieler, der in der vorangegangenen Runde als letztes den Einsatz erhöht hat. Die Berechnung für die Anzahl von Nodes, bei denen noch kein Spieler ausgestiegen ist, ist trivial.

$$\text{3PG-3P-PF-Con} = \text{Spieler mit der letzen Erhöhung} \times \text{Zahl unterschiedlicher Einsätze} = 3 * 4 = 12 \quad (5.1)$$

Es gibt 3 mögliche Spieler mit der letzten Erhöhung und die Spieler haben alle gleich viele Einsätze gebracht. Nach den Regeln für die Preflop-Runde sind das zwischen einem Einsatz und 4 Einsätzen, da 2 Spieler schon Einsätze gebracht haben und damit eine Runde, bei der alle Spieler noch im Spiel sind und kein Spieler Einsätze gebracht hat nicht möglich ist. Damit gibt es 4 mögliche Einsätze, mit denen die Runde enden kann und 12 unterscheidbare Continuing Nodes für die Preflop-Runde. Bei der Berechnung der Anzahl der Continuing Nodes müssen wir zusätzlich berücksichtigen, dass es mehrere Möglichkeiten gibt, nach welchem Einsatz der Spieler in der Setzrunde ausgestiegen ist und mehrere Möglichkeiten, 2 Spieler aus den 3 Spielern des Spieles auszuwählen. Diese Möglichkeiten können wir mit Hilfe kombinatorischer Formeln bestimmen. Sei n die Anzahl aller Spieler im Spiel, k die Anzahl der noch aktiven Spieler nach der Runde, a die Anzahl ausgestiegener Spieler, i die möglichen Einsätze der noch aktiven Spieler, r die Anzahl möglicher Erhöhungen in der aktuellen Setzrunde und m die kleinste mögliche Erhöhung in der Setzrunde. Dann gilt für die Anzahl möglichen Preflop Continuing Nodes:

$$\begin{aligned} \text{3PG-2P-X-Con} &= \binom{n}{k} * k * \sum_{i=1}^{r} i^a \\ \text{3PG-2P-X-Con} &= \binom{3}{2} * k * \sum_{i=1}^{4} i^1 \\ \text{3PG-2P-X-Con} &= 6 * (1^1 + 2^1 + 3^1 + 4^1) = 60 \end{aligned} \quad (5.2)$$

Die Anzahl möglicher Spieler aus der Menge aller Spieler zu ziehen, lösen wir mit der Binomialformel, da die Reihenfolge, in der die Spieler gezogen werden, nicht berücksichtigt werden muss. Jeder der gezogenen Spieler kann der Spieler mit der letzten Erhöhung der vorangegangenen Runde sein. Deshalb wird die Anzahl der Möglichkeiten noch mit der Anzahl gezogener Spieler multipliziert. Auch die Einsätze der ausgestiegenen Spieler müssen für den Spielzustand berücksichtigt werden. Diese können eine beliebige Anzahl von Bets eingesetzt haben, so lange diese geringer sind als die Anzahl der Bets der noch aktiven Spieler. Haben die noch aktiven Spieler in dieser Runde eine Bet eingesetzt, so können die ausgeschiedenen Spieler keine Bets eingesetzt haben. Bei zwei Bets der aktiven Spieler können Spieler mit keiner oder einer eingesetzten Bet das Spiel verlassen haben. Auch diese Anzahl der Möglichkeiten bestimmen wir wieder mit einer kombinatorischen Formel. In diesem Fall handelt es sich um eine Ziehung mit Wiederholungen, da mehrere Spieler, mit der gleichen Anzahl von gebrachten Einsätzen, ausgestiegen sein können. Die Reihenfolge der Ziehung ist hier wichtig, denn wir brauchen die Unterscheidung, welcher Spieler wie viele Bets eingesetzt hat, um später für die Utility-Funktionen der jeweiligen Spieler die exakten Verluste zurückgeben zu können.

5.2 Ergebnisse

In Tabelle 5.1 können wir die Reduktion der Continuing Nodes im Vergleich zum unabstrahierten 3-Spieler-Spiel ablesen. Die größere Anzahl von Zuständen in der Preflop-Runde mit IBR erklärt sich dadurch, dass wir in der Formel für die Berechnung außer Acht gelassen haben, dass zwei der Spieler bereits einen Einsatz in der ersten Runde gebracht haben und die Zahl ihrer möglichen Aktionen

Tabelle 5.1.: Continuing Nodes mit und ohne IBR

3-Player-Game-IBR	Preflop	Flop	Turn
Total Continuing Nodes	72	75	75
2-Player Continuing Nodes	60	60	60
3-Player Continuing Nodes	12	15	15
3-Player-Game	Preflop	Flop	Turn
Total Continuing Nodes	81	169	169
2-Player Continuing Nodes	59	123	123
3-Player Continuing Nodes	22	46	46

deshalb eingeschränkt ist. Das heißt, wir generieren Zustände, die in unserem Spielbaum eigentlich nicht vorkommen. Damit unterschätzen wir die tatsächlich mögliche Reduktion. In der Praxis sollten diese Folgezustände dann allerdings nicht im Spielbaum eingehängt werden und damit wegfallen. Auf den ersten Blick reduziert sich das Gesamtproblem auch hier nicht besonders stark. Dabei muss man aber berücksichtigen, dass die Continuing Nodes in die Berechnung der River-Runde, die maßgeblich die Gesamtgröße des Spielbaumes bestimmt, gleich dreifach als Faktor mit eingehen und damit die Größe des Spielbaums deutlich senken, wie in Tabelle 5.2 ersichtlich ist.

Tabelle 5.2.: Information Sets und Game States von 3-Spieler Fixed Limit Poker mit und ohne IBR

Kartenkombinationen	Runde	3-Player Game States	3-Player-IBR Game States
3 Players	Preflop	$1,56 * 10^{14}$	$1,56 * 10^{14}$
	Flop	$1,09 * 10^{20}$	$6,66 * 10^{19}$
	Turn	$2,27 * 10^{23}$	$4,75 * 10^{22}$
	River	$3,91 * 10^{26}$	$2,81 * 10^{25}$
	Total	$3,91 * 10^{26}$	$2,82 * 10^{25}$
Kartenkombinationen	Runde	3-Player Infosets	3-Player-IBSR Infosets
1 Player	Preflop	$1,09 * 10^{05}$	$1,09 * 10^{05}$
	Flop	$1,13 * 10^{11}$	$6,89 * 10^{10}$
	Turn	$2,69 * 10^{14}$	$5,63 * 10^{13}$
	River	$5,34 * 10^{17}$	$3,84 * 10^{16}$
	Total	$5,34 * 10^{17}$	$3,84 * 10^{16}$
1 Canonical Player	Preflop	$1,39 * 10^{04}$	$1,39 * 10^{04}$
	Flop	$5,60 * 10^{09}$	$3,41 * 10^{09}$
	Turn	$1,22 * 10^{13}$	$2,54 * 10^{12}$
	River	$2,31 * 10^{16}$	$1,66 * 10^{15}$
	Total	$2,31 * 10^{16}$	$1,66 * 10^{15}$
1 Bucketing Player	Preflop	$1,64 * 10^{02}$	$1,64 * 10^{02}$
	Flop	$1,74 * 10^{04}$	$1,06 * 10^{04}$
	Turn	$1,76 * 10^{06}$	$3,69 * 10^{05}$
	River	$1,52 * 10^{08}$	$1,09 * 10^{07}$
	Total	$1,54 * 10^{08}$	$1,13 * 10^{07}$

Tabelle 5.3.: Infoset Actions und Game State Actions von 3-Spieler Fixed Limit Poker mit und ohne IBR

Kartenkombinationen	Runde	3-Player Game State Actions	3-Player-IBR Game State Actions
3 Players	Preflop	$5,86 * 10^{14}$	$5,86 * 10^{14}$
	Flop	$3,95 * 10^{20}$	$1,65 * 10^{20}$
	Turn	$7,99 * 10^{23}$	$1,18 * 10^{23}$
	River	$1,40 * 10^{27}$	$7,01 * 10^{25}$
	Total	$1,40 * 10^{27}$	$7,02 * 10^{25}$
Kartenkombinationen	Runde	3-Player Infoset Actions	3-Player-IBR Infoset Actions
1 Player	Preflop	$4,10 * 10^{05}$	$4,10 * 10^{05}$
	Flop	$4,09 * 10^{11}$	$1,71 * 10^{11}$
	Turn	$9,48 * 10^{14}$	$1,40 * 10^{14}$
	River	$1,91 * 10^{18}$	$9,56 * 10^{16}$
	Total	$1,91 * 10^{18}$	$9,57 * 10^{16}$
1 Canonical Player	Preflop	$5,22 * 10^{04}$	$5,22 * 10^{04}$
	Flop	$2,02 * 10^{10}$	$8,45 * 10^{09}$
	Turn	$4,28 * 10^{13}$	$6,34 * 10^{12}$
	River	$8,26 * 10^{16}$	$4,13 * 10^{15}$
	Total	$8,27 * 10^{16}$	$4,14 * 10^{15}$
1 Bucketing Player	Preflop	$6,18 * 10^{02}$	$6,18 * 10^{02}$
	Flop	$6,29 * 10^{04}$	$2,63 * 10^{04}$
	Turn	$6,21 * 10^{06}$	$9,19 * 10^{05}$
	River	$5,44 * 10^{08}$	$2,72 * 10^{07}$
	Total	$5,51 * 10^{08}$	$2,82 * 10^{07}$

Tabelle 5.4.: Speicherbedarf mit und ohne IBR

Kartenkombinationen	Speicher ohne IBR	Speicher mit IBR
1 Player	$3,06 * 10^{10}$	$1,53 * 10^{09}$
1 Canonical Player	$1,32 * 10^{09}$	$6,62 * 10^{07}$
1 Bucketing Player	$8,81 * 10^{00}$	$4,51 * 10^{-01}$

6 Anwendung der Abstraktionen bei steigender Spielerzahl

Da sich die beiden untersuchten Abstraktionsvarianten speziell für Mehrspielervarianten eignen, ist es von entscheidender Bedeutung, wie sich die Abstraktionen bei steigenden Spielerzahlen verhalten. Am besten eignen sich dafür Abstraktionen, bei denen sich der Reduktionsfaktor bei steigender Spielerzahl weiter erhöht.

6.1 Komplexität von 4-Spieler Fixed Limit Poker

Um die Entwicklung der Abstraktionen bei steigenden Spielerzahlen zu untersuchen, ist es zuerst nötig, die Komplexität einer Variante mit mehr Spielern detalliert zu bestimmen. Deshalb beginne ich mit der Berechnung der Komplexität der 4-Spieler-Variante von Fixed-Limit Poker. Dazu werden die gleichen Formeln wie bei der Berechnung für den 3-Spieler-Fall verwendet.

6.1.1 Ergebnisse

Da die Anzahl der Knoten des Setzbaumes genau wie im 3-Spieler-Fall zu groß ist, um sie in einer Tabelle innerhalb der Arbeit gut darstellen zu können, befinden sich die übrigen Berechnungen wieder im Anhang. Auf die Historien wurde aus Platzgründen verzichtet. Der Setzbaum wurde mit Hilfe des gleichen Programmes wie im 3-Spieler-Fall erstellt. Die Aufstellungen der Anzahl unterschiedlicher Knotentypen des 4-Spieler-Spieles befindet sich in der Tabelle 6.1. Die Anzahl der Knoten erhöht sich auch beim Sprung von der 3- zur 4-Spieler-Variante um einen ähnlichen Faktor. Auch die Berechnung der Gesamtzahl von Knoten späterer Setzrunden ist analog zur 3-Spieler-Variante. Die Berechnung für die 2-Spieler-Sequence-Nodes ist als Beispiel in der Tabelle 6.2 dargestellt. Die Anzahl von Runden mit unterschiedlich vielen Aktiven Spielern an den Setzknoten an, sodass es zunehmen schwierig wird sicherzustellen, dass die Aufzählung vollständig ist. Wir können sie aber mit Hilfe der kombinatorischen Formel für eine Probe mit Wiederholungen ohne Beachtung der Reihenfolge überprüfen. Betrachten wir die möglichen Runden, in denen ein Spieler bis zur überprüften Runde aus dem Spiel aussteigen kann als n und die Anzahl der Runden in denen einer der Spieler aussteigt als k, so ziehen wir z.B. im Falle der River Runde und einem ausgestiegenen Spieler eine Runde aus 4 möglichen Runden oder

$$\binom{n+k-1}{k} = \binom{4}{1} = 4$$

Die Anzahl aller möglichen Sequence Nodes und die Anzahl aller möglichen Aktionen an den Sequence Nodes ist in den Tabellen 6.3 und 6.4 dargestellt. Zusammen mit den Kartenkombinationen aus der Tabelle 6.5 ergeben sich dann die Game States und Game State Actions, bzw. Informationsets und Informationset Actions in Tabelle 6.6.

6.2 Truncated Multiplayer Nodes in 4-Spieler-Fixed-Limit-Poker

Die Berechnung der Anzahl der Gesamtkonten erfolgt wie im 3-Spieler Fall, indem nur die 2-Spieler Knoten mit der Anzahl möglicher Kartenkombinationen der jeweiligen Runde verrechnet wird. Sei k^r die Anzahl möglicher Kartenkombinationen in der Flop Runde und $SeqN_2, s_3^N, \cdots, s_N^N$ die Gesamtzahl aller Sequence Nodes in der Runde r gilt dann:

$$Seq = c * Seq_2^N + \sum_{i=1}^{N} s_i^N \tag{6.1}$$

Tabelle 6.1.: Anzahl der Knoten des Setzbaumes von 4-Spieler-Fixed-Limit-Poker

Node Type	Preflop	Flop	Turn	River
Total Sequences	698	2.180	2.180	2.180
2-Player Sequences	193	616	616	616
3-Player Sequences	345	1.080	1.080	1.080
4-Player Sequences	160	484	484	484
Total Actions	1.587	4.972	4.972	4.972
2-Player Actions	433	1.388	1.388	1.388
3-Player Actions	783	2.460	2.460	2.460
4-Player Actions	371	1.124	1.124	1.124
Total Continuing Nodes	697	2.177	2.177	0
2-Player Continuing Nodes	392	1.236	1.236	0
3-Player Continuing Nodes	252	780	780	0
4-Player Continuing Nodes	53	161	161	0
Total Terminal Nodes	193	616	616	2.793

Tabelle 6.2.: Berechnung der Knoten des Setzbaumes von 4-Spieler-Fixed-Limit-Poker

Preflop	
4P-2P-PF-Seq	193
Flop	
4P-4P-PF-Con * 4P-2P-F-Seq	32.648
4P-3P-PF-Con * 3P-2P-F-Seq	19.656
4P-2P-PF-Con * 2P-2P-F-Seq	3.920
Total	56.224
Turn	
4P-4P-PF-Con * 4P-4P-F-Con * 4P-2P-T-Seq	5.256.328
4P-4P-PF-Con * 4P-3P-F-Con * 3P-2P-T-Seq	3.224.520
4P-3P-PF-Con * 3P-3P-F-Con * 3P-2P-T-Seq	904.176
4P-3P-PF-Con * 3P-2P-F-Con * 2P-2P-T-Seq	309.960
4P-2P-PF-Con * 2P-2P-F-Con * 2P-2P-T-Seq	35.280
4P-4P-PF-Con * 4P-2P-F-Con * 2P-2P-T-Seq	655.080
Total	9.730.264
River	
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Con * 4P-2P-R-Seq	846.268.808
4P-4P-PF-Con * 4P-4P-F-Con * 4P-3P-T-Con * 3P-2P-R-Seq	519.147.720
4P-4P-PF-Con * 4P-4P-F-Con * 4P-2P-T-Con * 2P-2P-R-Seq	105.467.880
4P-4P-PF-Con * 4P-3P-F-Con * 3P-3P-T-Con * 3P-2P-R-Seq	148.327.920
4P-4P-PF-Con * 4P-3P-F-Con * 3P-2P-T-Con * 2P-2P-R-Seq	50.848.200
4P-4P-PF-Con * 4P-2P-F-Con * 2P-2P-T-Con * 2P-2P-R-Seq	5.895.720
4P-3P-PF-Con * 3P-3P-F-Con * 3P-3P-T-Con * 3P-2P-R-Seq	41.592.096
4P-3P-PF-Con * 3P-3P-F-Con * 3P-2P-T-Con * 2P-2P-R-Seq	14.258.160
4P-3P-PF-Con * 3P-2P-F-Con * 2P-2P-T-Con * 2P-2P-R-Seq	2.789.640
4P-2P-PF-Con * 2P-2P-F-Con * 2P-2P-T-Con * 2P-2P-R-Seq	317.520
Total	1.734.913.664

Tabelle 6.3.: Sequence Nodes von 4-Spieler-Fixed-Limit-Poker

Round	Total	2-Player	3-Player	4-Player
Preflop	$6,98 * 10^{02}$	$1,93 * 10^{02}$	$3,45 * 10^{02}$	$1,60 * 10^{02}$
Flop	$1,63 * 10^{05}$	$5,62 * 10^{04}$	$8,07 * 10^{04}$	$2,57 * 10^{04}$
Turn	$2,80 * 10^{07}$	$9,73 * 10^{06}$	$1,41 * 10^{07}$	$4,13 * 10^{06}$
River	$4,73 * 10^{09}$	$1,73 * 10^{09}$	$2,33 * 10^{09}$	$6,65 * 10^{08}$
Total	$4,76 * 10^{09}$	$1,74 * 10^{09}$	$2,34 * 10^{09}$	$6,69 * 10^{08}$

Tabelle 6.4.: Sequence Actions von 4-Spieler-Fixed-Limit-Poker

Round	Total	2-Player	3-Player	4-Player
Preflop	$1,59 * 10^{03}$	$4,33 * 10^{02}$	$7,83 * 10^{02}$	$3,71 * 10^{02}$
Flop	$3,79 * 10^{05}$	$1,31 * 10^{05}$	$1,88 * 10^{05}$	$5,96 * 10^{04}$
Turn	$6,54 * 10^{07}$	$2,27 * 10^{07}$	$3,31 * 10^{07}$	$9,59 * 10^{06}$
River	$1,11 * 10^{10}$	$4,09 * 10^{09}$	$5,45 * 10^{09}$	$1,54 * 10^{09}$
Total	$1,12 * 10^{10}$	$4,11 * 10^{09}$	$5,49 * 10^{09}$	$1,55 * 10^{09}$

Tabelle 6.5.: Kartenkombinationen der Spielertypen für 4-Spieler-Fixed-Limit-Poker

Round	4-Player	1-Player	Canonical-1-Player	Card Bucketing
Preflop	$1,90 * 10^{12}$	$1,33 * 10^{03}$	$1,69 * 10^{02}$	2
Flop	$2,51 * 10^{16}$	$2,60 * 10^{07}$	$1,29 * 10^{06}$	4
Turn	$1,03 * 10^{18}$	$1,22 * 10^{09}$	$5,52 * 10^{07}$	8
River	$4,12 * 10^{19}$	$5,62 * 10^{10}$	$2,43 * 10^{09}$	16

Tabelle 6.6.: Game States und Information Sets von 4-Spieler-Fixed-Limit-Poker

Kartenkombinationen	Runde	Game States	Game State Actions
4 Players	Preflop	$1,32 * 10^{15}$	$3,01 * 10^{15}$
	Flop	$4,08 * 10^{21}$	$9,51 * 10^{21}$
	Turn	$2,88 * 10^{25}$	$6,73 * 10^{25}$
	River	$1,95 * 10^{29}$	$4,57 * 10^{29}$
	Total	$1,95 * 10^{29}$	$4,57 * 10^{29}$
Kartenkombinationen	Runde	Infosets	Infoset-Actions
1 Player	Preflop	$9,26 * 10^{05}$	$2,10 * 10^{06}$
	Flop	$4,22 * 10^{12}$	$9,84 * 10^{12}$
	Turn	$3,42 * 10^{16}$	$7,99 * 10^{16}$
	River	$2,66 * 10^{20}$	$6,23 * 10^{20}$
	Total	$2,66 * 10^{20}$	$6,23 * 10^{20}$
1 Canonical Player	Preflop	$1,18 * 10^{05}$	$2,68 * 10^{05}$
	Flop	$2,09 * 10^{11}$	$4,87 * 10^{11}$
	Turn	$1,55 * 10^{15}$	$3,61 * 10^{15}$
	River	$1,15 * 10^{19}$	$2,69 * 10^{19}$
	Total	$1,15 * 10^{19}$	$2,69 * 10^{19}$
1 Bucketing Player	Preflop	$1,40 * 10^{03}$	$3,17 * 10^{03}$
	Flop	$6,50 * 10^{05}$	$1,52 * 10^{06}$
	Turn	$2,24 * 10^{08}$	$5,23 * 10^{08}$
	River	$7,57 * 10^{10}$	$1,77 * 10^{11}$
	Total	$7,59 * 10^{10}$	$1,78 * 10^{11}$

Tabelle 6.7.: Information Sets und Game States von 4-Spieler Fixed Limit Poker mit TMN

Kartenkombinationen	Runde	Game States	Game State Actions
4 Players	Preflop	$3,66 * 10^{14}$	$8,21 * 10^{14}$
	Flop	$1,41 * 10^{21}$	$3,30 * 10^{21}$
	Turn	$1,00 * 10^{25}$	$2,34 * 10^{25}$
	River	$7,15 * 10^{28}$	$1,69 * 10^{29}$
	Total	$7,15 * 10^{28}$	$1,69 * 10^{29}$
Kartenkombinationen	Runde	Infosets	Infoset-Actions
1 Player	Preflop	$2,56 * 10^{05}$	$5,75 * 10^{05}$
	Flop	$1,46 * 10^{12}$	$3,41 * 10^{12}$
	Turn	$1,19 * 10^{16}$	$2,78 * 10^{16}$
	River	$9,75 * 10^{19}$	$2,30 * 10^{20}$
	Total	$9,75 * 10^{19}$	$2,30 * 10^{20}$
1 Canonical Player	Preflop	$3,31 * 10^{04}$	$7,43 * 10^{04}$
	Flop	$7,23 * 10^{10}$	$1,69 * 10^{11}$
	Turn	$5,37 * 10^{14}$	$1,26 * 10^{15}$
	River	$4,21 * 10^{18}$	$9,94 * 10^{18}$
	Total	$4,21 * 10^{18}$	$9,94 * 10^{18}$
1 Bucketing Player	Preflop	$8,91 * 10^{02}$	$2,02 * 10^{03}$
	Flop	$3,31 * 10^{05}$	$7,73 * 10^{05}$
	Turn	$9,61 * 10^{07}$ l	$2,25 * 10^{08}$
	River	$3,08 * 10^{10}$	$7,25 * 10^{10}$
	Total	$3,08 * 10^{1}0$	$7,27 * 10^{10}$

6.2.1 Komplexitätsreduktion in Abhändigkeit zur Spielerzahl

Intuitiv wurde angenommen, dass sich durch diese Abstraktion die Anzahl der Knoten um einen größeren Faktor, im Vergleich zum 3-Spieler-Spiel, reduziert. Wenn wir die Anzahl von 2-Spieler-Knoten im Vergleich zur Gesamtzahl von Sequence Nodes in der Tabellen 6.3 betrachten ist zu erkennen, dass das Verhältnis mit $36,55\%$ etwas mehr als $\frac{1}{3}$ aller Knoten ausmacht. Im 3-Spieler-Spiel lag der Anteil noch bei $54,42\%$ und damit grob beim Faktor $\frac{1}{2}$. Die Entwicklung des Anteils von 2-Spieler-Knoten in den Mehrspieler-Varianten lässt vermuten, dass der Anteil von 2-Spieler-Knoten sich auch bei steigender Spielerzahl um den Faktor $\frac{1}{N-1}$ bewegt und damit linear mit der steigenden Spielerzahl sinkt. Da die Anzahl von Kartenkombinationen identisch mit denen der unabstrahierten 4-Spieler-Variante sind und das Verhältnis von Information Sets zu Sequence Nodes damit gleich bleibt können wir daraus schließen, dass sich die Größe des Spielbaumes ungefähr um $N-1$ reduziert. Da die Größe des Spielbaumes bei steigender Spielerzahl allerdings exponentiell zunimmt, kann diese Abstraktion auch bei steigenden Spielerzahlen die Größe des Problems nicht ausreichend reduzieren, um signifikant stärkere Strategien zu berechnen.

6.3 Imperfect Betting Recall in 4-Spieler Fixed Limit Poker

Wie auch beim 2-Player-Node-Focus, können wir die Formeln zur Berechnung der Information Sets direkt aus der 3-Spieler-Variante übernehmen. Sei n die Anzahl aller Spieler im Spiel, k die Anzahl der noch aktiven Spieler nach der Runde, a die Anzahl ausgestiegener Spieler, i die möglichen Einsätze der

Tabelle 6.8.: Continuing Nodes von 4-Spieler-Fixed-Limit-Poker mit IBR

4PG-4P-X-Con	4PG-3P-X-Con	4PG-2P-X-Con
20	120	360

noch aktiven Spieler, r die Anzahl möglicher Erhöhungen in der aktuellen Setzrunde und m die kleinste mögliche Erhöhung in der Setzrunde. Dann gilt für die Anzahl möglichen Preflop Continuing Nodes:

$$\begin{aligned} \text{4PG-2P-X-Con} &= \binom{n}{k} * k * \sum_{i=1}^{r} i^a \\ \text{4PG-2P-X-Con} &= \binom{4}{2} * k * \sum_{i=1}^{4} i^2 \\ \text{4PG-2P-X-Con} &= 12 * (1^2 + 2^2 + 3^2 + 4^2) = 360 \end{aligned} \tag{6.2}$$

Die so berechneten Continuing Nodes für alle Spielrunden mit 4 aktiven Spielern sind in Tabelle 6.8 zusammengefasst.

Daraus ergeben sich für die 4-Spieler Variante die Anzahl an Sequence Nodes und Sequence Actions in den Tabellen 6.9 und 6.10. Die Game States und Information Sets sind in der Tabelle 6.11 zusammengefasst. Die Zahl an Sequence Nodes reduziert sich durch die Abstraktion also von $4,76 * 10^{09}$ auf $4,37 * 10^{07}$, d.h. um den Faktor $108,9$. Vergleichen wir das mit dem Faktor, um den sich die Anzahl der Sequence Nodes im 3-Spieler-Spiel reduziert hat, $9,72 * 10^{06} \div 7,32 * 10^{05} = 13,2787$, zeigt sich, dass der Faktor sich mit steigender Spielerzahl deutlich erhöht. Auch wenn wir die Faktoren im Verhältnis zur Spielerzahl betrachten ist die Reduktion im 4-Spieler Spiel noch wesentlich größer.

$$108,9 \div 4 = 27,225 > 13,2787 \div 3 = 4,4262$$

Der Faktor, um den sich der Spielbaum durch die Abstraktion verkleinert, scheint sich also mit steigender Spielerzahl exponentiell zu erhöhen. Um den Faktor für beliebige Spielerzahlen grob anzunähern bestimmen wir die Wachstumsrate der Exponentialfunktion in Abhängigkeit unserer Spielerzahl. Sei $B(0)$ der Reduktionsfaktor von IBR für die 3-Spieler-Variante und $B(1)$ der Reduktionsfaktor für 4 Spieler, dann gilt für die Wachstumsrate der Exponentialfunktion:

$$p = (B(0)/div B(1)) - 1 = (108,9/div 13,2787) - 1 = 720,11\%$$

Mit Hilfe der Wachstumsrate können wir die Exponentialfunktion annähern:

$$\begin{aligned} B(t) &= B(0) * e^{\lambda * t} \\ \lambda &= ln(1+p) = 2,1043 \\ B(t) &= 13,2787 * e^{2,1043 * t} \end{aligned} \tag{6.3}$$

Die Problemgröße reduziert sich bei steigender Spielerzahl also ebenfalls exponenziell wodurch die Abstraktion sich für Mehrspieler-Varianten gut eignet. Besonders interessant ist, dass es mit einem sehr groben Card Bucketing von 2 Buckets je Runde, theoretisch bereits möglich wäre eine Lösung für 4-Spieler-Fixed-Limit-Poker berechnen, wie aus Tabelle 6.12 ersichtlich wird.

Tabelle 6.9.: Sequence Nodes von 4-Spieler-Fixed-Limit-Poker mit IBR

Round	Total	2-Player	3-Player	4-Player
Preflop	$6,98 * 10^{02}$	$1,93 * 10^{02}$	$3,45 * 10^{02}$	$1,60 * 10^{02}$
Flop	$6,77 * 10^{04}$	$2,53 * 10^{04}$	$3,28 * 10^{04}$	$9,68 * 10^{03}$
Turn	$1,69 * 10^{06}$	$6,78 * 10^{05}$	$8,23 * 10^{05}$	$1,94 * 10^{05}$
River	$4,20 * 10^{07}$	$1,91 * 10^{07}$	$1,90 * 10^{07}$	$3,87 * 10^{06}$
Total	$4,37 * 10^{07}$	$1,98 * 10^{07}$	$1,98 * 10^{07}$	$4,08 * 10^{06}$

Tabelle 6.10.: Sequence Actions von 4-Spieler-Fixed-Limit-Poker mit IBR

Round	Total	2-Player	3-Player	4-Player
Preflop	$1,59 * 10^{03}$	$4,33 * 10^{02}$	$7,83 * 10^{02}$	$3,71 * 10^{02}$
Flop	$1,59 * 10^{05}$	$5,98 * 10^{04}$	$7,66 * 10^{04}$	$2,25 * 10^{04}$
Turn	$4,01 * 10^{06}$	$1,62 * 10^{06}$	$1,94 * 10^{06}$	$4,50 * 10^{05}$
River	$1,00 * 10^{08}$	$4,65 * 10^{07}$	$4,50 * 10^{07}$	$8,99 * 10^{06}$
Total	$1,05 * 10^{08}$	$4,82 * 10^{07}$	$4,70 * 10^{07}$	$9,46 * 10^{06}$

Tabelle 6.11.: Game States und Information Sets von 4-Spieler-Fixed-Limit-Poker

Kartenkombinationen	Runde	Game States	Game State Actions
4 Players	Preflop	$1,32 * 10^{15}$	$3,01 * 10^{15}$
	Flop	$1,70 * 10^{21}$	$3,99 * 10^{21}$
	Turn	$1,75 * 10^{24}$	$4,13 * 10^{24}$
	River	$1,73 * 10^{27}$	$4,14 * 10^{27}$
	Total	$1,73 * 10^{27}$	$4,14 * 10^{27}$
Kartenkombinationen	Runde	Infosets	Infoset-Actions
1 Player	Preflop	$9,26 * 10^{05}$	$2,10 * 10^{06}$
	Flop	$1,76 * 10^{12}$	$4,13 * 10^{12}$
	Turn	$2,07 * 10^{15}$	$4,90 * 10^{15}$
	River	$2,36 * 10^{18}$	$5,65 * 10^{18}$
	Total	$2,36 * 10^{18}$	$5,65 * 10^{18}$
1 Canonical Player	Preflop	$1,18 * 10^{05}$	$2,68 * 10^{05}$
	Flop	$8,71 * 10^{10}$	$2,04 * 10^{11}$
	Turn	$9,35 * 10^{13}$	$2,21 * 10^{14}$
	River	$1,02 * 10^{17}$	$2,44 * 10^{17}$
	Total	$1,02 * 10^{17}$	$2,44 * 10^{17}$
1 Bucketing Player	Preflop	$1,40 * 10^{03}$	$3,17 * 10^{03}$
	Flop	$2,71 * 10^{05}$	$6,35 * 10^{05}$
	Turn	$1,36 * 10^{07}$	$3,21 * 10^{07}$
	River	$6,71 * 10^{08}$	$1,61 * 10^{09}$
	Total	$6,85 * 10^{0}8$	$1,64 * 10^{09}$

Tabelle 6.12.: Speicherbedarf von 4-Spieler-Fixed-Limit-Poker

Kartenkombinationen	Speicher in GB
1 Player	90.419.307.516, 79
1 Canonical Player	3.907.717.506, 38
1 Bucketing Player	26, 25

7 Fazit

Die Bestimmung der Komplexität von Multiplayer Fixed-Limit Poker hat interessante Einblicke über den Abstraktionsgrad der beiden untersuchten Abstraktionen, sowie die Zusammensetzung der Setzbäume ermöglicht. Bei der Berechnung der Komplexität des 3-Spieler-Fixed-Limit-Spielbaumes hat sich die Annahme, die 2-Spieler-Knoten würden einen relativ geringen Anteil der Gesamtknoten ausmachen, leider nicht bestätigt. Die 2-Spieler-Knoten sind mit $54,41\%$ sogar häufiger als 3-Spieler Knoten. Dazu kommt der recht hohe Informationsverlust und die Tatsache, dass der CFR-Algorithmus, trotz fehlender theoretischer Garantien für die Existenz eines nash-Gleichgewichts, auch für Mehrspieler-Varianten gute Strategien berechnen kann. Unter diesen Vorraussetzungen kann ich mir kein Szenario vorstellen, in dem TMN sinnvoll einsetzbar wäre.
Erfreulich sind die sehr viel besseren Ergebnisse von IBR, bei dessen Entwicklung die Struktur des Setzbaumes eine entscheidende Rolle gespielt hat. Dass der ohnehin schon bessere Reduktionsfaktor mit mit steigender Spielerzahl exponentiell ansteigt, macht IBR zu einer interessanten Variante, gerade im Hinblick auf die Lösbarkeit von Multiplayer Fixed-Limit Poker. Mit der kleinstmöglichen Kartenabstraktion könnte theoretisch bereits eine Strategie mit Hilfe des CFR Algorithmus für die 4-Spieler-Variante berechnet werden.

Literatur

[Coma] *Annual Computer Poker Competition 2016 Rules*. URL: http://www.computerpokercompetition.org/index.php/competitions/rules/112-2016-rules.

[Comb] *Annual Computer Poker Competition 2017 Rules*. URL: http://www.computerpokercompetition.org/index.php/competitions/rules/119-2017-rules.

[Dön] Jan Dönges. *Das Go-Spiel ist geknackt*. URL: http://www.spektrum.de/news/das-go-spiel-ist-geknackt/1397114.

[Gib14] Richard Gibson. "Regret Minimization in Games and the Development of Champion Multiplayer Computer Poker-Playing Agents". Diss. 2014.

[GSS07] Andrew Gilpin, Tuomas Sandholm und Troels Bjerre Sørensen. "Potential-Aware Automated Abstraction of Sequential Games, and Holistic Equilibrium Analysis of Texas Hold'em Poker." In: *AAAI*. AAAI Press, 2007, S. 50–57. ISBN: 978-1-57735-323-2. URL: http://dblp.uni-trier.de/db/conf/aaai/aaai2007.html#GilpinSS07; http://www.aaai.org/Library/AAAI/2007/aaai07-008.php; http://www.bibsonomy.org/bibtex/2d9f60871ba76f32147999ee4d4eaaf6a/dblp.

[Hol] *Texas Hold'em*. Juli 2016. URL: https://de.wikipedia.org/wiki/Texas_Hold%E2%80%99em.

[Joh+13] Michael Johanson u. a. "Evaluating state-space abstractions in extensive-form games." In: *AAMAS*. Hrsg. von Maria L. Gini u. a. IFAAMAS, 2013, S. 271–278. ISBN: 978-1-4503-1993-5. URL: http://dblp.uni-trier.de/db/conf/atal/aamas2013.html#JohansonBVB13; http://dl.acm.org/citation.cfm?id=2484965;http://www.bibsonomy.org/bibtex/24b73bc02ede478599513d11e988bc9c6/dblp.

[Joh13] Michael Johanson. "Measuring the Size of Large No-Limit Poker Games". In: *CoRR* abs/1302.7008 (2013).

[Joh16] Michael Bradley Johanson. "Robust Strategies and Counter-Strategies: From Superhuman to Optimal Play". Diss. 2016.

[OR94] M. Osborne und A. Rubenstein. *A Course in Game Theory*. Cambridge, MA: MIT Press, 1994.

[Ris] Nicholas Abou Risk. "Using Counterfactual Regret Minimization to Create a Competitive Mutliplayer Poker Agent". Magisterarb.

[Wau+10] Kevin Waugh u. a. "A Practical Use of Imperfect Recall." In: *SARA*. Hrsg. von Vadim Bulitko und J. Christopher Beck. AAAI, 9. Mai 2010. URL: http://dblp.uni-trier.de/db/conf/sara/sara2009.html#WaughZJKSB09; http://www.aaai.org/ocs/index.php/SARA/SARA09/paper/view/839; http://www.bibsonomy.org/bibtex/273ca484a7e6596a6a45a5fe6777470ad/dblp.

A Komplexität von 3-Spieler-Fixed-Limit-Poker

A.1 Sequences

A.1.1 Aufzählung aller Sequences

Die Aktionen bestehen jeweils aus einem Kürzel und dem Spieler in Klammern, der die Aktion durchführt. C steht für Check/Call, f für Fold und r für Raise.

Preflop

Number of 2-player Sequences = 37
(c(P0), c(P1), r(P2), c(P0), r(P1), c(P2), r(P0), f(P1))
(c(P0), c(P1), r(P2), c(P0), r(P1), f(P2))
(c(P0), c(P1), r(P2), c(P0), r(P1), f(P2), r(P0))
(c(P0), c(P1), r(P2), c(P0), r(P1), r(P2), f(P0))
(c(P0), c(P1), r(P2), f(P0))
(c(P0), c(P1), r(P2), f(P0), r(P1))
(c(P0), c(P1), r(P2), f(P0), r(P1), r(P2))
(c(P0), c(P1), r(P2), r(P0), c(P1), r(P2), f(P0))
(c(P0), c(P1), r(P2), r(P0), f(P1))
(c(P0), c(P1), r(P2), r(P0), f(P1), r(P2))
(c(P0), c(P1), r(P2), r(P0), r(P1), f(P2))
(c(P0), f(P1))
(c(P0), f(P1), r(P2))
(c(P0), f(P1), r(P2), r(P0))
(c(P0), f(P1), r(P2), r(P0), r(P2))
(c(P0), r(P1), c(P2), r(P0), c(P1), r(P2), f(P0))
(c(P0), r(P1), c(P2), r(P0), f(P1))
(c(P0), r(P1), c(P2), r(P0), f(P1), r(P2))
(c(P0), r(P1), c(P2), r(P0), r(P1), f(P2))
(c(P0), r(P1), f(P2))
(c(P0), r(P1), f(P2), r(P0))
(c(P0), r(P1), f(P2), r(P0), r(P1))
(c(P0), r(P1), r(P2), c(P0), r(P1), f(P2))
(c(P0), r(P1), r(P2), f(P0))
(c(P0), r(P1), r(P2), f(P0), r(P1))
(c(P0), r(P1), r(P2), r(P0), f(P1))
(r(P0), c(P1), r(P2), c(P0), r(P1), f(P2))
(r(P0), c(P1), r(P2), f(P0))
(r(P0), c(P1), r(P2), f(P0), r(P1))
(r(P0), c(P1), r(P2), r(P0), f(P1))
(r(P0), f(P1))
(r(P0), f(P1), r(P2))
(r(P0), f(P1), r(P2), r(P0))
(r(P0), r(P1), c(P2), r(P0), f(P1))

(r(P0), r(P1), f(P2))
(r(P0), r(P1), f(P2), r(P0))
(r(P0), r(P1), r(P2), f(P0))

Number of 3-player Sequences = 45

()
(c(P0))
(c(P0), c(P1))
(c(P0), c(P1), r(P2))
(c(P0), c(P1), r(P2), c(P0))
(c(P0), c(P1), r(P2), c(P0), r(P1))
(c(P0), c(P1), r(P2), c(P0), r(P1), c(P2))
(c(P0), c(P1), r(P2), c(P0), r(P1), c(P2), r(P0))
(c(P0), c(P1), r(P2), c(P0), r(P1), c(P2), r(P0), c(P1))
(c(P0), c(P1), r(P2), c(P0), r(P1), r(P2))
(c(P0), c(P1), r(P2), c(P0), r(P1), r(P2), c(P0))
(c(P0), c(P1), r(P2), r(P0))
(c(P0), c(P1), r(P2), r(P0), c(P1))
(c(P0), c(P1), r(P2), r(P0), c(P1), r(P2))
(c(P0), c(P1), r(P2), r(P0), c(P1), r(P2), c(P0))
(c(P0), c(P1), r(P2), r(P0), r(P1))
(c(P0), c(P1), r(P2), r(P0), r(P1), c(P2))
(c(P0), r(P1))
(c(P0), r(P1), c(P2))
(c(P0), r(P1), c(P2), r(P0))
(c(P0), r(P1), c(P2), r(P0), c(P1))
(c(P0), r(P1), c(P2), r(P0), c(P1), r(P2))
(c(P0), r(P1), c(P2), r(P0), c(P1), r(P2), c(P0))
(c(P0), r(P1), c(P2), r(P0), r(P1))
(c(P0), r(P1), c(P2), r(P0), r(P1), c(P2))
(c(P0), r(P1), r(P2))
(c(P0), r(P1), r(P2), c(P0))
(c(P0), r(P1), r(P2), c(P0), r(P1))
(c(P0), r(P1), r(P2), c(P0), r(P1), c(P2))
(c(P0), r(P1), r(P2), r(P0))
(c(P0), r(P1), r(P2), r(P0), c(P1))
(r(P0))
(r(P0), c(P1))
(r(P0), c(P1), r(P2))
(r(P0), c(P1), r(P2), c(P0))
(r(P0), c(P1), r(P2), c(P0), r(P1))
(r(P0), c(P1), r(P2), c(P0), r(P1), c(P2))
(r(P0), c(P1), r(P2), r(P0))
(r(P0), c(P1), r(P2), r(P0), c(P1))
(r(P0), r(P1))
(r(P0), r(P1), c(P2))
(r(P0), r(P1), c(P2), r(P0))
(r(P0), r(P1), c(P2), r(P0), c(P1))
(r(P0), r(P1), r(P2))

(r(P0), r(P1), r(P2), c(P0))

Flop/Turn/River

Number of 2-player Sequences = 78
(c(P0), c(P1), r(P2), c(P0), r(P1), c(P2), r(P0), c(P1), r(P2), f(P0))
(c(P0), c(P1), r(P2), c(P0), r(P1), c(P2), r(P0), f(P1))
(c(P0), c(P1), r(P2), c(P0), r(P1), c(P2), r(P0), f(P1), r(P2))
(c(P0), c(P1), r(P2), c(P0), r(P1), c(P2), r(P0), r(P1), f(P2))
(c(P0), c(P1), r(P2), c(P0), r(P1), f(P2))
(c(P0), c(P1), r(P2), c(P0), r(P1), f(P2), r(P0))
(c(P0), c(P1), r(P2), c(P0), r(P1), f(P2), r(P0), r(P1))
(c(P0), c(P1), r(P2), c(P0), r(P1), r(P2), c(P0), r(P1), f(P2))
(c(P0), c(P1), r(P2), c(P0), r(P1), r(P2), f(P0))
(c(P0), c(P1), r(P2), c(P0), r(P1), r(P2), f(P0), r(P1))
(c(P0), c(P1), r(P2), c(P0), r(P1), r(P2), r(P0), f(P1))
(c(P0), c(P1), r(P2), f(P0))
(c(P0), c(P1), r(P2), f(P0), r(P1))
(c(P0), c(P1), r(P2), f(P0), r(P1), r(P2))
(c(P0), c(P1), r(P2), f(P0), r(P1), r(P2), r(P1))
(c(P0), c(P1), r(P2), r(P0), c(P1), r(P2), c(P0), r(P1), f(P2))
(c(P0), c(P1), r(P2), r(P0), c(P1), r(P2), f(P0))
(c(P0), c(P1), r(P2), r(P0), c(P1), r(P2), f(P0), r(P1))
(c(P0), c(P1), r(P2), r(P0), c(P1), r(P2), r(P0), f(P1))
(c(P0), c(P1), r(P2), r(P0), f(P1))
(c(P0), c(P1), r(P2), r(P0), f(P1), r(P2))
(c(P0), c(P1), r(P2), r(P0), f(P1), r(P2), r(P0))
(c(P0), c(P1), r(P2), r(P0), r(P1), c(P2), r(P0), f(P1))
(c(P0), c(P1), r(P2), r(P0), r(P1), f(P2))
(c(P0), c(P1), r(P2), r(P0), r(P1), f(P2), r(P0))
(c(P0), c(P1), r(P2), r(P0), r(P1), r(P2), f(P0))
(c(P0), r(P1), c(P2), r(P0), c(P1), r(P2), c(P0), r(P1), f(P2))
(c(P0), r(P1), c(P2), r(P0), c(P1), r(P2), f(P0))
(c(P0), r(P1), c(P2), r(P0), c(P1), r(P2), f(P0), r(P1))
(c(P0), r(P1), c(P2), r(P0), c(P1), r(P2), r(P0), f(P1))
(c(P0), r(P1), c(P2), r(P0), f(P1))
(c(P0), r(P1), c(P2), r(P0), f(P1), r(P2))
(c(P0), r(P1), c(P2), r(P0), f(P1), r(P2), r(P0))
(c(P0), r(P1), c(P2), r(P0), r(P1), c(P2), r(P0), f(P1))
(c(P0), r(P1), c(P2), r(P0), r(P1), f(P2))
(c(P0), r(P1), c(P2), r(P0), r(P1), f(P2), r(P0))
(c(P0), r(P1), c(P2), r(P0), r(P1), r(P2), f(P0))
(c(P0), r(P1), f(P2))
(c(P0), r(P1), f(P2), r(P0))
(c(P0), r(P1), f(P2), r(P0), r(P1))
(c(P0), r(P1), f(P2), r(P0), r(P1), r(P0))
(c(P0), r(P1), r(P2), c(P0), r(P1), c(P2), r(P0), f(P1))
(c(P0), r(P1), r(P2), c(P0), r(P1), f(P2))
(c(P0), r(P1), r(P2), c(P0), r(P1), f(P2), r(P0))

(c(P0), r(P1), r(P2), c(P0), r(P1), r(P2), f(P0))
(c(P0), r(P1), r(P2), f(P0))
(c(P0), r(P1), r(P2), f(P0), r(P1))
(c(P0), r(P1), r(P2), f(P0), r(P1), r(P2))
(c(P0), r(P1), r(P2), r(P0), c(P1), r(P2), f(P0))
(c(P0), r(P1), r(P2), r(P0), f(P1))
(c(P0), r(P1), r(P2), r(P0), f(P1), r(P2))
(c(P0), r(P1), r(P2), r(P0), r(P1), f(P2))
(r(P0), c(P1), r(P2), c(P0), r(P1), c(P2), r(P0), f(P1))
(r(P0), c(P1), r(P2), c(P0), r(P1), f(P2))
(r(P0), c(P1), r(P2), c(P0), r(P1), f(P2), r(P0))
(r(P0), c(P1), r(P2), c(P0), r(P1), r(P2), f(P0))
(r(P0), c(P1), r(P2), f(P0))
(r(P0), c(P1), r(P2), f(P0), r(P1))
(r(P0), c(P1), r(P2), f(P0), r(P1), r(P2))
(r(P0), c(P1), r(P2), r(P0), c(P1), r(P2), f(P0))
(r(P0), c(P1), r(P2), r(P0), f(P1))
(r(P0), c(P1), r(P2), r(P0), f(P1), r(P2))
(r(P0), c(P1), r(P2), r(P0), r(P1), f(P2))
(r(P0), f(P1))
(r(P0), f(P1), r(P2))
(r(P0), f(P1), r(P2), r(P0))
(r(P0), f(P1), r(P2), r(P0), r(P2))
(r(P0), r(P1), c(P2), r(P0), c(P1), r(P2), f(P0))
(r(P0), r(P1), c(P2), r(P0), f(P1))
(r(P0), r(P1), c(P2), r(P0), f(P1), r(P2))
(r(P0), r(P1), c(P2), r(P0), r(P1), f(P2))
(r(P0), r(P1), f(P2))
(r(P0), r(P1), f(P2), r(P0))
(r(P0), r(P1), f(P2), r(P0), r(P1))
(r(P0), r(P1), r(P2), c(P0), r(P1), f(P2))
(r(P0), r(P1), r(P2), f(P0))
(r(P0), r(P1), r(P2), f(P0), r(P1))
(r(P0), r(P1), r(P2), r(P0), f(P1))

Number of 3-player Sequences = 93 ()
(c(P0))
(c(P0), c(P1))
(c(P0), c(P1), r(P2))
(c(P0), c(P1), r(P2), c(P0))
(c(P0), c(P1), r(P2), c(P0), r(P1))
(c(P0), c(P1), r(P2), c(P0), r(P1), c(P2))
(c(P0), c(P1), r(P2), c(P0), r(P1), c(P2), r(P0))
(c(P0), c(P1), r(P2), c(P0), r(P1), c(P2), r(P0), c(P1))
(c(P0), c(P1), r(P2), c(P0), r(P1), c(P2), r(P0), c(P1), r(P2))
(c(P0), c(P1), r(P2), c(P0), r(P1), c(P2), r(P0), c(P1), r(P2), c(P0))
(c(P0), c(P1), r(P2), c(P0), r(P1), c(P2), r(P0), r(P1))
(c(P0), c(P1), r(P2), c(P0), r(P1), c(P2), r(P0), r(P1), c(P2))
(c(P0), c(P1), r(P2), c(P0), r(P1), r(P2))
(c(P0), c(P1), r(P2), c(P0), r(P1), r(P2), c(P0))

(c(P0), c(P1), r(P2), c(P0), r(P1), r(P2), c(P0), r(P1))
(c(P0), c(P1), r(P2), c(P0), r(P1), r(P2), c(P0), r(P1), c(P2))
(c(P0), c(P1), r(P2), c(P0), r(P1), r(P2), r(P0))
(c(P0), c(P1), r(P2), c(P0), r(P1), r(P2), r(P0), c(P1))
(c(P0), c(P1), r(P2), r(P0))
(c(P0), c(P1), r(P2), r(P0), c(P1))
(c(P0), c(P1), r(P2), r(P0), c(P1), r(P2))
(c(P0), c(P1), r(P2), r(P0), c(P1), r(P2), c(P0))
(c(P0), c(P1), r(P2), r(P0), c(P1), r(P2), c(P0), r(P1))
(c(P0), c(P1), r(P2), r(P0), c(P1), r(P2), c(P0), r(P1), c(P2))
(c(P0), c(P1), r(P2), r(P0), c(P1), r(P2), r(P0))
(c(P0), c(P1), r(P2), r(P0), c(P1), r(P2), r(P0), c(P1))
(c(P0), c(P1), r(P2), r(P0), r(P1))
(c(P0), c(P1), r(P2), r(P0), r(P1), c(P2))
(c(P0), c(P1), r(P2), r(P0), r(P1), c(P2), r(P0))
(c(P0), c(P1), r(P2), r(P0), r(P1), c(P2), r(P0), c(P1))
(c(P0), c(P1), r(P2), r(P0), r(P1), r(P2))
(c(P0), c(P1), r(P2), r(P0), r(P1), r(P2), c(P0))
(c(P0), r(P1))
(c(P0), r(P1), c(P2))
(c(P0), r(P1), c(P2), r(P0))
(c(P0), r(P1), c(P2), r(P0), c(P1))
(c(P0), r(P1), c(P2), r(P0), c(P1), r(P2))
(c(P0), r(P1), c(P2), r(P0), c(P1), r(P2), c(P0))
(c(P0), r(P1), c(P2), r(P0), c(P1), r(P2), c(P0), r(P1))
(c(P0), r(P1), c(P2), r(P0), c(P1), r(P2), c(P0), r(P1), c(P2))
(c(P0), r(P1), c(P2), r(P0), c(P1), r(P2), r(P0))
(c(P0), r(P1), c(P2), r(P0), c(P1), r(P2), r(P0), c(P1))
(c(P0), r(P1), c(P2), r(P0), r(P1))
(c(P0), r(P1), c(P2), r(P0), r(P1), c(P2))
(c(P0), r(P1), c(P2), r(P0), r(P1), c(P2), r(P0))
(c(P0), r(P1), c(P2), r(P0), r(P1), c(P2), r(P0), c(P1))
(c(P0), r(P1), c(P2), r(P0), r(P1), r(P2))
(c(P0), r(P1), c(P2), r(P0), r(P1), r(P2), c(P0))
(c(P0), r(P1), r(P2))
(c(P0), r(P1), r(P2), c(P0))
(c(P0), r(P1), r(P2), c(P0), r(P1))
(c(P0), r(P1), r(P2), c(P0), r(P1), c(P2))
(c(P0), r(P1), r(P2), c(P0), r(P1), c(P2), r(P0))
(c(P0), r(P1), r(P2), c(P0), r(P1), c(P2), r(P0), c(P1))
(c(P0), r(P1), r(P2), c(P0), r(P1), r(P2))
(c(P0), r(P1), r(P2), c(P0), r(P1), r(P2), c(P0))
(c(P0), r(P1), r(P2), r(P0))
(c(P0), r(P1), r(P2), r(P0), c(P1))
(c(P0), r(P1), r(P2), r(P0), c(P1), r(P2))
(c(P0), r(P1), r(P2), r(P0), c(P1), r(P2), c(P0))
(c(P0), r(P1), r(P2), r(P0), r(P1))
(c(P0), r(P1), r(P2), r(P0), r(P1), c(P2))
(r(P0))
(r(P0), c(P1))

Tabelle A.1.: Sequence Nodes von 3-Spieler-Fixed-Limit-Poker

Round	Total Sequences	2-Player Sequences	3-Player Sequences
Preflop	82	37	45
Flop	4.352	2.306	2.046
Turn	220.602	126.486	94.116
River	9.496.482	5.167.146	4.329.336
Total	9.721.518	5.295.975	4.425.543

(r(P0), c(P1), r(P2))
(r(P0), c(P1), r(P2), c(P0))
(r(P0), c(P1), r(P2), c(P0), r(P1))
(r(P0), c(P1), r(P2), c(P0), r(P1), c(P2))
(r(P0), c(P1), r(P2), c(P0), r(P1), c(P2), r(P0))
(r(P0), c(P1), r(P2), c(P0), r(P1), c(P2), r(P0), c(P1))
(r(P0), c(P1), r(P2), c(P0), r(P1), r(P2))
(r(P0), c(P1), r(P2), c(P0), r(P1), r(P2), c(P0))
(r(P0), c(P1), r(P2), r(P0))
(r(P0), c(P1), r(P2), r(P0), c(P1))
(r(P0), c(P1), r(P2), r(P0), c(P1), r(P2))
(r(P0), c(P1), r(P2), r(P0), c(P1), r(P2), c(P0))
(r(P0), c(P1), r(P2), r(P0), r(P1))
(r(P0), c(P1), r(P2), r(P0), r(P1), c(P2))
(r(P0), r(P1))
(r(P0), r(P1), c(P2))
(r(P0), r(P1), c(P2), r(P0))
(r(P0), r(P1), c(P2), r(P0), c(P1))
(r(P0), r(P1), c(P2), r(P0), c(P1), r(P2))
(r(P0), r(P1), c(P2), r(P0), c(P1), r(P2), c(P0))
(r(P0), r(P1), c(P2), r(P0), r(P1))
(r(P0), r(P1), c(P2), r(P0), r(P1), c(P2))
(r(P0), r(P1), r(P2))
(r(P0), r(P1), r(P2), c(P0))
(r(P0), r(P1), r(P2), c(P0), r(P1))
(r(P0), r(P1), r(P2), c(P0), r(P1), c(P2))
(r(P0), r(P1), r(P2), r(P0))
(r(P0), r(P1), r(P2), r(P0), c(P1))

A.1.2 Ohne Imperfect Betting Recall

Anzahl von 3-Player-Sequences

Preflop
3PG-3P-PF-Seq = 45
Flop
3PG-3P-PF-Con * 3P-3P-F-Seq = 2.046
Turn
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-3P-T-Seq = 94.116

Tabelle A.2.: Sequence Nodes von 3-Spieler-Fixed-Limit-Poker

Round	Total Sequences	2-Player Sequences	3-Player Sequences
Preflop	82	37	45
Flop	2.652	1.536	1.116
Turn	46.080	29.340	16.740
River	683.100	432.000	251.100
Total	731.914	462.913	269.001

River
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-3P-T-Con * 3PG-3P-R-Seq = 4.329.336

Anzahl von 2-Player-Sequences

Preflop
3PG-2P-PF-Seq = 37
Flop
3PG-3P-PF-Con * 3PG-2P-F-Seq = 1.716
3PG-2P-PF-Con * 2PG-2P-F-Seq = 590
Total = 2306
Turn
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-2P-T-Seq = 94.116
3PG-3P-PF-Con * 3PG-2P-F-Con * 2PG-2P-T-Seq = 27.060
3PG-2P-PF-Con * 2PG-2P-F-Con * 2PG-2P-T-Seq = 5.310
Total =126.486
River
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-3P-T-Con * 3PG-2P-R-Seq = 3.631.056
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-2P-T-Con * 2PG-2P-R-Seq = 1.244.760
3PG-3P-PF-Con * 3PG-2P-F-Con * 2PG-2P-T-Con * 2PG-2P-R-Seq = 243.540
3PG-2P-PF-Con * 2PG-2P-F-Con * 2PG-2P-T-Con * 2PG-2P-R-Seq = 47.790
Total = 5.167.146

A.1.3 Mit Imperfect Betting Recall

Anzahl von 3-Player-Sequences

Preflop
3PG-3P-PF-Seq = 45
Flop
3PG-3P-PF-Con * 3P-3P-F-Seq = 1.116
Turn
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-3P-T-Seq = 16.740
River
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-3P-T-Con * 3PG-3P-R-Seq = 251.100

Tabelle A.3.: Sequence Actions von 3-Spieler-Fixed-Limit-Poker

Round	Total Sequence Actions	2-Player-Sequence-Actions	3-Player-Sequence-Actions
Preflop	309	89	110
Flop	15.724	5.692	5.016
Turn	776.370	314.898	230.736
River	34.019.874	12.792.162	10.613.856
Total	34.812.277	13.112.841	10.849.718

Anzahl von 2-Player-Sequences

Preflop
3PG-2P-PF-Seq = 37
Flop
3PG-3P-PF-Con * 3PG-2P-F-Seq = 936
3PG-2P-PF-Con * 2PG-2P-F-Seq = 600
Total = 1.536
Turn
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-2P-T-Seq = 16.740
3PG-3P-PF-Con * 3PG-2P-F-Con * 2PG-2P-T-Seq = 7.200
3PG-2P-PF-Con * 2PG-2P-F-Con * 2PG-2P-T-Seq = 5.400
Total = 29.340
River
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-3P-T-Con * 3PG-2P-R-Seq = 210.600
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-2P-T-Con * 2PG-2P-R-Seq = 108.000
3PG-3P-PF-Con * 3PG-2P-F-Con * 2PG-2P-T-Con * 2PG-2P-R-Seq = 64.800
3PG-2P-PF-Con * 2PG-2P-F-Con * 2PG-2P-T-Con * 2PG-2P-R-Seq = 48.600
Total = 432.000

A.2 Sequence Actions

A.2.1 Ohne Imperfect Betting Recall

3-Player-Sequence-Actions

Preflop
3PG-3P-PF-SeqAct = 110
Flop
3PG-3P-PF-Con * 3P-3P-F-SeqAct = 5.016
Turn
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-3P-T-SeqAct = 230.736
River
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-3P-T-Con * 3PG-3P-R-SeqAct = 10.613.856

Tabelle A.4.: Sequence Actions von 3-Spieler-Fixed-Limit-Poker

Round	Total Sequence Actions	2-Player-Sequence-Actions	3-Player-Sequence-Actions
Preflop	309	89	110
Flop	6.564	3.828	2.736
Turn	114.840	73.800	41.040
River	1.701.540	1.085.940	615.600
Total	1.823.253	1.163.657	659.486

2-Player-Sequence-Actions

Preflop
3PG-2P-PF-SeqAct = 89
Flop
3PG-3P-PF-Con * 3PG-2P-F-SeqAct = 4.158
3PG-2P-PF-Con * 2PG-2P-F-SeqAct = 1.534
Total = 5.692
Turn
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-2P-T-SeqAct = 230.736
3PG-3P-PF-Con * 3PG-2P-F-Con * 2PG-2P-T-SeqAct = 70.356
3PG-2P-PF-Con * 2PG-2P-F-Con * 2PG-2P-T-SeqAct = 13.806
Total = 314.898
River
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-3P-T-Con * 3PG-2P-R-SeqAct = 8.798.328
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-2P-T-Con * 2PG-2P-R-SeqAct = 3.236.376
3PG-3P-PF-Con * 3PG-2P-F-Con * 2PG-2P-T-Con * 2PG-2P-R-SeqAct = 633.204
3PG-2P-PF-Con * 2PG-2P-F-Con * 2PG-2P-T-Con * 2PG-2P-R-SeqAct = 124.254
Total = 12.792.162

A.2.2 Mit Imperfect Betting Recall

3-Player-Sequence-Actions

Preflop
3PG-3P-PF-SeqAct = 110
Flop
3PG-3P-PF-Con * 3P-3P-F-SeqAct = 2.736
Turn
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-3P-T-SeqAct = 41.040
River
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-3P-T-Con * 3PG-3P-R-SeqAct = 615.600

2-Player-Sequence-Actions

Preflop
3PG-2P-PF-Seq = 89

Tabelle A.5.: Continuing Nodes von 3-Spieler-Fixed-Limit-Poker

Round	Total Continuing Nodes	2-Player Continuing Nodes	3-Player Continuing Nodes
Preflop	103	59	22
Flop	5.261	3.237	1.012
Turn	168.789	75.685	46.552
River	0	0	0
Total	174.153	78.981	47.586

Flop
3PG-3P-PF-Con * 3PG-2P-F-SeqAct = 2.268
3PG-2P-PF-Con * 2PG-2P-F-SeqAct = 1.560
Total = 3.828
Turn
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-2P-T-SeqAct = 41.040
3PG-3P-PF-Con * 3PG-2P-F-Con * 2PG-2P-T-SeqAct = 18.720
3PG-2P-PF-Con * 2PG-2P-F-Con * 2PG-2P-T-SeqAct = 14.040
Total = 73.800
River
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-3P-T-Con * 3PG-2P-R-SeqAct = 510.300
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-2P-T-Con * 2PG-2P-R-SeqAct = 280.800
3PG-3P-PF-Con * 3PG-2P-F-Con * 2PG-2P-T-Con * 2PG-2P-R-SeqAct = 168.480
3PG-2P-PF-Con * 2PG-2P-F-Con * 2PG-2P-T-Con * 2PG-2P-R-SeqAct = 126.360
Total = 1.085.940

A.3 Continuing Nodes

A.3.1 Ohne Imperfect Betting Recall

3-Player Continuing Nodes

Preflop
3PG-3P-PF-Con = 22
Flop
3PG-3P-PF-Con * 3P-3P-F-Con = 1.012
Turn
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-3P-T-Con = 46.552

2-Player Continuing Nodes

Preflop
3PG-2P-PF-Con = 59
Flop
3PG-3P-PF-Con * 3PG-2P-F-Con = 2.706
3PG-2P-PF-Con * 2PG-2P-F-Con = 531
Total = 3.237

Tabelle A.6.: Continuing Nodes von 3-Spieler-Fixed-Limit-Poker

Round	Total Continuing Nodes	2-Player Continuing Nodes	3-Player Continuing Nodes
Preflop	72	60	12
Flop	1.440	1.260	180
Turn	16.740	14.040	2.700
River	0	0	0
Total	18.252	15.360	2.892

Turn
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-2P-T-Con = 46.552
3PG-3P-PF-Con * 3PG-2P-F-Con * 2PG-2P-T-Con = 24.354
3PG-2P-PF-Con * 2PG-2P-F-Con * 2PG-2P-T-Con = 4.779
Total = 75.685

A.3.2 Mit Imperfect Betting Recall

3-Player Continuing Nodes

Preflop
3PG-3P-PF-Con = 12
Flop
3PG-3P-PF-Con * 3P-3P-F-Con = 180
Turn
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-3P-T-Con = 2700

2-Player Continuing Nodes

Preflop
3PG-2P-PF-Con = 60
Flop
3PG-3P-PF-Con * 3PG-2P-F-Con = 720
3PG-2P-PF-Con * 2PG-2P-F-Con = 540
Total = 1.260
Turn
3PG-3P-PF-Con * 3PG-3P-F-Con * 3PG-2P-T-Con = 2.700
3PG-3P-PF-Con * 3PG-2P-F-Con * 2PG-2P-T-Con = 6.480
3PG-2P-PF-Con * 2PG-2P-F-Con * 2PG-2P-T-Con = 4.860
Total = 14.040

B Komplexität von 4-Spieler-Fixed-Limit-Poker

B.1 Sequences

B.1.1 Ohne Imperfect Betting Recall

Anzahl von 4-Player-Sequences

Preflop
4P-4P-PF-Seq = 160
Flop
4P-4P-PF-Con * 4P-4P-F-Seq = 25652
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Seq = 4.129.972
River
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Con * 4P-4P-R-Seq = 664.925.492

Anzahl von 3-Player-Sequences

Preflop
4P-3P-PF-Seq = 345
Flop
4P-4P-PF-Con * 4P-3P-F-Seq = 57.240
4P-3P-PF-Con * 3P-3P-F-Seq = 23.436
Total = 80.676
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-3P-T-Seq = 9.215.640
4P-4P-PF-Con * 4P-3P-F-Con * 3P-3P-T-Seq = 3.844.620
4P-3P-PF-Con * 3P-3P-F-Con * 3P-3P-T-Seq = 1.078.056
Total = 14.138.316
River
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Con * 4P-3P-R-Seq = 1.483.718.040
4P-4P-PF-Con * 4P-4P-F-Con * 4P-3P-T-Con * 3P-3P-R-Seq = 618.983.820
4P-4P-PF-Con * 4P-3P-F-Con * 3P-3P-T-Con * 3P-3P-R-Seq = 176.852.520
4P-3P-PF-Con * 3P-3P-F-Con * 3P-3P-T-Con * 3P-3P-R-Seq = 49.590.576
Total = 2.329.144.956

Tabelle B.1.: Sequence Nodes von 4-Spieler-Fixed-Limit-Poker

Round	Total Sequences	2-Player-Sequences	3-Player-Sequences	4-Player-Sequences
Preflop	698	193	345	160
Flop	162.552 56.224	80.676	25.652	
Turn	27.998.552	9.730.264	14.138.316 4.129.972	
River	4.728.984.112	1.734.913.664	2.329.144.956	664.925.492
Total	4.757.145.914	1.744.700.345	2.343.364.293	669.081.276

Tabelle B.2.: Sequence Nodes von 4-Spieler-Fixed-Limit-Poker

Round	Total Sequences	2-Player-Sequences	3-Player-Sequences	4-Player-Sequences
Preflop	698	193	345	160
Flop	59.000	22.816	28.440	7.744
Turn	1.438.120	591.680	691.560	154.880
River	34.902.200	16.132.000	15.672.600	3.097.600
Total	36.400.018	16.746.689	16.392.945	3.260.384

Anzahl von 2-Player-Sequences

Preflop
4P-2P-PF-Seq = 193
Flop
4P-4P-PF-Con * 4P-2P-F-Seq = 32.648
4P-3P-PF-Con * 3P-2P-F-Seq = 19.656
4P-2P-PF-Con * 2P-2P-F-Seq = 3.920
Total = 56.224
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-2P-T-Seq = 5.256.328
4P-4P-PF-Con * 4P-3P-F-Con * 3P-2P-T-Seq = 3.224.520
4P-3P-PF-Con * 3P-3P-F-Con * 3P-2P-T-Seq = 904.176
4P-3P-PF-Con * 3P-2P-F-Con * 2P-2P-T-Seq = 309.960
4P-2P-PF-Con * 2P-2P-F-Con * 2P-2P-T-Seq = 35.280
4P-4P-PF-Con * 4P-2P-F-Con * 2P-2P-T-Seq = 655.080
Total = 9.730.264
River
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Con * 4P-2P-R-Seq = 846.268.808
4P-4P-PF-Con * 4P-4P-F-Con * 4P-3P-T-Con * 3P-2P-R-Seq = 519.147.720
4P-4P-PF-Con * 4P-4P-F-Con * 4P-2P-T-Con * 2P-2P-R-Seq = 105.467.880
4P-4P-PF-Con * 4P-3P-F-Con * 3P-3P-T-Con * 3P-2P-R-Seq = 148.327.920
4P-4P-PF-Con * 4P-3P-F-Con * 3P-2P-T-Con * 2P-2P-R-Seq = 50.848.200
4P-4P-PF-Con * 4P-2P-F-Con * 2P-2P-T-Con * 2P-2P-R-Seq = 5.895.720
4P-3P-PF-Con * 3P-3P-F-Con * 3P-3P-T-Con * 3P-2P-R-Seq = 41.592.096
4P-3P-PF-Con * 3P-3P-F-Con * 3P-2P-T-Con * 2P-2P-R-Seq = 14.258.160
4P-3P-PF-Con * 3P-2P-F-Con * 2P-2P-T-Con * 2P-2P-R-Seq = 2.789.640
4P-2P-PF-Con * 2P-2P-F-Con * 2P-2P-T-Con * 2P-2P-R-Seq = 317.520
Total = 1.734.913.664

B.1.2 Mit Imperfect Betting Recall

Anzahl von 4-Player-Sequences

Preflop
4P-4P-PF-Seq = 160
Flop
4P-4P-PF-Con * 4P-4P-F-Seq = 7.744
Turn

4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Seq = 154.880
River
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Con * 4P-4P-R-Seq = 3.097.600

Anzahl von 3-Player-Sequences

Preflop
4P-3P-PF-Seq = 345
Flop
4P-4P-PF-Con * 4P-3P-F-Seq = 17.280
4P-3P-PF-Con * 3P-3P-F-Seq = 11.160
Total = 28.440
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-3P-T-Seq = 345.600
4P-4P-PF-Con * 4P-3P-F-Con * 3P-3P-T-Seq = 178.560
4P-3P-PF-Con * 3P-3P-F-Con * 3P-3P-T-Seq = 167.400
Total = 691.560
River
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Con * 4P-3P-R-Seq = 6.912.000
4P-4P-PF-Con * 4P-4P-F-Con * 4P-3P-T-Con * 3P-3P-R-Seq = 3.571.200
4P-4P-PF-Con * 4P-3P-F-Con * 3P-3P-T-Con * 3P-3P-R-Seq = 2.678.400
4P-3P-PF-Con * 3P-3P-F-Con * 3P-3P-T-Con * 3P-3P-R-Seq = 2.511.000
Total = 15.672.600

Anzahl von 2-Player-Sequences

Preflop
4P-2P-PF-Seq = 193
Flop
4P-4P-PF-Con * 4P-2P-F-Seq = 9.856
4P-3P-PF-Con * 3P-2P-F-Seq = 9.360
4P-2P-PF-Con * 2P-2P-F-Seq = 3.600
Total = 22.816
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-2P-T-Seq = 197.120
4P-4P-PF-Con * 4P-3P-F-Con * 3P-2P-T-Seq = 149.760
4P-3P-PF-Con * 3P-3P-F-Con * 3P-2P-T-Seq = 140.400
4P-3P-PF-Con * 3P-2P-F-Con * 2P-2P-T-Seq = 72.000
4P-2P-PF-Con * 2P-2P-F-Con * 2P-2P-T-Seq = 32.400
4P-4P-PF-Con * 4P-2P-F-Con * 2P-2P-T-Seq = 57.600
Total = 591.680
River
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Con * 4P-2P-R-Seq = 3.942.400
4P-4P-PF-Con * 4P-4P-F-Con * 4P-3P-T-Con * 3P-2P-R-Seq = 2.995.200
4P-4P-PF-Con * 4P-4P-F-Con * 4P-2P-T-Con * 2P-2P-R-Seq = 1.152.000
4P-4P-PF-Con * 4P-3P-F-Con * 3P-3P-T-Con * 3P-2P-R-Seq = 2.246.400
4P-4P-PF-Con * 4P-3P-F-Con * 3P-2P-T-Con * 2P-2P-R-Seq = 1.152.000
4P-4P-PF-Con * 4P-2P-F-Con * 2P-2P-T-Con * 2P-2P-R-Seq = 518.400

Tabelle B.3.: Sequence Actions von 4-Spieler-Fixed-Limit-Poker

Round	Total Sequence-Actions	2-Player-Sequence-Actions	3-Player-Sequence-Actions	4-Player-Sequence
Preflop	1.587	433	783	371
Flop	378.792	131.384	187.836	59.572
Turn	65.396.344	22.745.576	33.059.676 9.591.092	
River	11.088.283.952	4.091.878.624	5.452.239.516	1.544.165.812
Total	11.154.060.675	4.114.756.017	5.485.487.811	1.553.816.847

4P-3P-PF-Con * 3P-3P-F-Con * 3P-3P-T-Con * 3P-2P-R-Seq = 2.106.000
4P-3P-PF-Con * 3P-3P-F-Con * 3P-2P-T-Con * 2P-2P-R-Seq = 1.080.000
4P-3P-PF-Con * 3P-2P-F-Con * 2P-2P-T-Con * 2P-2P-R-Seq = 648.000
4P-2P-PF-Con * 2P-2P-F-Con * 2P-2P-T-Con * 2P-2P-R-Seq = 291.600
Total = 16.132.000

B.2 Sequence Actions

B.2.1 Ohne Imperfect Betting Recall

Anzahl von 4-Player-Sequence-Actions

Preflop
4P-4P-PF-SeqAct = 371
Flop
4P-4P-PF-Con * 4P-4P-F-SeqAct = 59.572
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-SeqAct = 9.591.092
River
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Con * 4P-4P-R-SeqAct = 1.544.165.812

Anzahl von 3-Player-Sequence-Actions

Preflop
4P-3P-PF-SeqAct = 783
Flop
4P-4P-PF-Con * 4P-3P-F-SeqAct = 130.380
4P-3P-PF-Con * 3P-3P-F-SeqAct = 57.456
Total = 187.836
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-3P-T-SeqAct = 20.991.180
4P-4P-PF-Con * 4P-3P-F-Con * 3P-3P-T-SeqAct = 9.425.520
4P-3P-PF-Con * 3P-3P-F-Con * 3P-3P-T-SeqAct = 2.642.976
Total = 33.059.676
River
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Con * 4P-3P-R-SeqAct = 3.379.579.980
4P-4P-PF-Con * 4P-4P-F-Con * 4P-3P-T-Con * 3P-3P-R-SeqAct = 1.517.508.720

Tabelle B.4.: Sequence Nodes von 4-Spieler-Fixed-Limit-Poker

Round	Total Sequence-Actions	2-Player-Sequence-Actions	3-Player-Sequence-Actions	4-Player-Sequence
Preflop	1.587	433	783	371
Flop	138.952	54.248 66.720	17.984	
Turn	3.413.720	1.418.680	1.635.360	359.680
River	83.691.400	39.276.200	37.221.600	7.193.600
Total	87.245.659	40.749.561	38.924.463	7.571.635

4P-4P-PF-Con * 4P-3P-F-Con * 3P-3P-T-Con * 3P-3P-R-SeqAct = 433.573.920
4P-3P-PF-Con * 3P-3P-F-Con * 3P-3P-T-Con * 3P-3P-R-SeqAct = 121.576.896
Total = 5.452.239.516

Anzahl von 2-Player-Sequence-Actions

Preflop
4P-2P-PF-SeqAct = 433
Flop
4P-4P-PF-Con * 4P-2P-F-SeqAct = 73.564
4P-3P-PF-Con * 3P-2P-F-SeqAct = 47.628
4P-2P-PF-Con * 2P-2P-F-SeqAct = 10.192
Total = 131.384
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-2P-T-SeqAct = 11.843.804
4P-4P-PF-Con * 4P-3P-F-Con * 3P-2P-T-SeqAct = 7.813.260
4P-3P-PF-Con * 3P-3P-F-Con * 3P-2P-T-SeqAct = 2.190.888
4P-3P-PF-Con * 3P-2P-F-Con * 2P-2P-T-SeqAct = 805.896
4P-2P-PF-Con * 2P-2P-F-Con * 2P-2P-T-SeqAct = 91.728
4P-4P-PF-Con * 4P-2P-F-Con * 2P-2P-T-SeqAct = 1.703.208
Total = 22.745.576
River
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Con * 4P-2P-R-SeqAct = 1.906.852.444
4P-4P-PF-Con * 4P-4P-F-Con * 4P-3P-T-Con * 3P-2P-R-SeqAct = 1.257.934.860
4P-4P-PF-Con * 4P-4P-F-Con * 4P-2P-T-Con * 2P-2P-R-SeqAct = 274.216.488
4P-4P-PF-Con * 4P-3P-F-Con * 3P-3P-T-Con * 3P-2P-R-SeqAct = 359.409.960
4P-4P-PF-Con * 4P-3P-F-Con * 3P-2P-T-Con * 2P-2P-R-SeqAct = 132.205.320
4P-4P-PF-Con * 4P-2P-F-Con * 2P-2P-T-Con * 2P-2P-R-SeqAct = 15.328.872
4P-3P-PF-Con * 3P-3P-F-Con * 3P-3P-T-Con * 3P-2P-R-SeqAct = 100.780.848
4P-3P-PF-Con * 3P-3P-F-Con * 3P-2P-T-Con * 2P-2P-R-SeqAct = 37.071.216
4P-3P-PF-Con * 3P-2P-F-Con * 2P-2P-T-Con * 2P-2P-R-SeqAct = 7.253.064
4P-2P-PF-Con * 2P-2P-F-Con * 2P-2P-T-Con * 2P-2P-R-SeqAct = 825.552
Total = 4.091.878.624

B.2.2 Mit Imperfect Betting Recall

Anzahl von 4-Player-Sequence-Actions

Preflop
4P-4P-PF-SeqAct = 371
Flop
4P-4P-PF-Con * 4P-4P-F-SeqAct = 17.984
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-SeqAct = 359.680
River
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Con * 4P-4P-R-SeqAct = 7.193.600

Anzahl von 3-Player-Sequence-Actions

Preflop
4P-3P-PF-SeqAct = 783
Flop
4P-4P-PF-Con * 4P-3P-F-SeqAct = 39.360
4P-3P-PF-Con * 3P-3P-F-SeqAct = 27.360
Total = 66.720
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-3P-T-SeqAct = 787.200
4P-4P-PF-Con * 4P-3P-F-Con * 3P-3P-T-SeqAct = 437.760
4P-3P-PF-Con * 3P-3P-F-Con * 3P-3P-T-SeqAct = 410.400
Total = 1.635.360
River
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Con * 4P-3P-R-SeqAct = 15.744.000
4P-4P-PF-Con * 4P-4P-F-Con * 4P-3P-T-Con * 3P-3P-R-SeqAct = 8.755.200
4P-4P-PF-Con * 4P-3P-F-Con * 3P-3P-T-Con * 3P-3P-R-SeqAct = 6.566.400
4P-3P-PF-Con * 3P-3P-F-Con * 3P-3P-T-Con * 3P-3P-R-SeqAct = 6.156.000
Total = 37.221.600

Anzahl von 2-Player-Sequence-Actions

Preflop
4P-2P-PF-SeqAct = 433
Flop
4P-4P-PF-Con * 4P-2P-F-SeqAct = 22.208
4P-3P-PF-Con * 3P-2P-F-SeqAct = 22.680
4P-2P-PF-Con * 2P-2P-F-SeqAct = 9.360
Total = 54.248
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-2P-T-SeqAct = 444.160
4P-4P-PF-Con * 4P-3P-F-Con * 3P-2P-T-SeqAct = 362.880
4P-3P-PF-Con * 3P-3P-F-Con * 3P-2P-T-SeqAct = 340.200
4P-3P-PF-Con * 3P-2P-F-Con * 2P-2P-T-SeqAct = 187.200
4P-2P-PF-Con * 2P-2P-F-Con * 2P-2P-T-SeqAct = 84.240
4P-4P-PF-Con * 4P-2P-F-Con * 2P-2P-T-SeqAct = 149.760
Total = 1.418.680

Tabelle B.5.: Continuing Nodes von 4-Spieler-Fixed-Limit-Poker

Round	Total Continuing Nodes	2-Player Continuing Nodes	3-Player Continuing Nodes	4-Player Continu
Preflop	697	392	252	53
Flop	161.497	100.032	52.932	8.533
Turn	27.832.565	17.368.140	9.090.612	1.373.813
River	0	0	0	0
Total	27.994.759	17.468.564	9.143.796	1.382.399

River
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Con * 4P-2P-R-SeqAct = 8.883.200
4P-4P-PF-Con * 4P-4P-F-Con * 4P-3P-T-Con * 3P-2P-R-SeqAct = 7.257.600
4P-4P-PF-Con * 4P-4P-F-Con * 4P-2P-T-Con * 2P-2P-R-SeqAct = 2.995.200
4P-4P-PF-Con * 4P-3P-F-Con * 3P-3P-T-Con * 3P-2P-R-SeqAct = 5.443.200
4P-4P-PF-Con * 4P-3P-F-Con * 3P-2P-T-Con * 2P-2P-R-SeqAct = 2.995.200
4P-4P-PF-Con * 4P-2P-F-Con * 2P-2P-T-Con * 2P-2P-R-SeqAct = 1.347.840
4P-3P-PF-Con * 3P-3P-F-Con * 3P-3P-T-Con * 3P-2P-R-SeqAct = 5.103.000
4P-3P-PF-Con * 3P-3P-F-Con * 3P-2P-T-Con * 2P-2P-R-SeqAct = 2.808.000
4P-3P-PF-Con * 3P-2P-F-Con * 2P-2P-T-Con * 2P-2P-R-SeqAct = 1.684.800
4P-2P-PF-Con * 2P-2P-F-Con * 2P-2P-T-Con * 2P-2P-R-SeqAct = 758.160
Total = 39.276.200

B.3 Continuing Nodes

B.3.1 Ohne Imperfect Betting Recall

Anzahl von 4-Player Continuing Nodes

Preflop
4P-4P-PF-Con = 53
Flop
4P-4P-PF-Con * 4P-4P-F-Con = 8.533
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Con = 1.373.813

Anzahl von 3-Player Continuing Nodes

Preflop
4P-3P-PF-Con = 252
Flop
4P-4P-PF-Con * 4P-3P-F-Con = 41.340
4P-3P-PF-Con * 3P-3P-F-Con = 11.592
Total = 52.932
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-3P-T-Con = 6.655.740
4P-4P-PF-Con * 4P-3P-F-Con * 3P-3P-T-Con = 1.901.640

Tabelle B.6.: Continuing Nodes von 4-Spieler-Fixed-Limit-Poker

Round	Total Continuing Nodes	2-Player Continuing Nodes	3-Player Continuing Nodes	4-Player Continu
Preflop	496	360	120	16
Flop	20.240	16.200	3.720	320
Turn	532.960	432.360	94.200	6.400
River	0	0	0	0
Total	553.696	448.920	98.040	6.736

4P-3P-PF-Con * 3P-3P-F-Con * 3P-3P-T-Con = 533.232
Total = 9.090.612

Anzahl von 2-Player Continuing Nodes

Preflop
4P-2P-PF-Con = 392
Flop
4P-4P-PF-Con * 4P-2P-F-Con = 65.508
4P-3P-PF-Con * 3P-2P-F-Con = 30.996
4P-2P-PF-Con * 2P-2P-F-Con = 3.528
Total = 100.032
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-2P-T-Con = 10.546.788
4P-4P-PF-Con * 4P-3P-F-Con * 3P-2P-T-Con = 5.084.820
4P-3P-PF-Con * 3P-3P-F-Con * 3P-2P-T-Con = 1.425.816
4P-3P-PF-Con * 3P-2P-F-Con * 2P-2P-T-Con = 278.964
4P-2P-PF-Con * 2P-2P-F-Con * 2P-2P-T-Con = 31.752
4P-4P-PF-Con * 4P-2P-F-Con * 2P-2P-T-Con = 589.572
Total = 17.368.140

B.3.2 Mit Imperfect Betting Recall

Anzahl von 4-Player Continuing Nodes

Preflop
4P-4P-PF-Con = 16
Flop
4P-4P-PF-Con * 4P-4P-F-Con = 320
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-4P-T-Con = 6.400

Anzahl von 3-Player Continuing Nodes

Preflop
4P-3P-PF-Con = 120

Flop
4P-4P-PF-Con * 4P-3P-F-Con = 1.920
4P-3P-PF-Con * 3P-3P-F-Con = 1.800
Total = 3.720
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-3P-T-Con = 38.400
4P-4P-PF-Con * 4P-3P-F-Con * 3P-3P-T-Con = 28.800
4P-3P-PF-Con * 3P-3P-F-Con * 3P-3P-T-Con = 27.000
Total = 94.200

Anzahl von 2-Player Continuing Nodes

Preflop
4P-2P-PF-Con = 360
Flop
4P-4P-PF-Con * 4P-2P-F-Con = 5.760
4P-3P-PF-Con * 3P-2P-F-Con = 7.200
4P-2P-PF-Con * 2P-2P-F-Con = 3.240
Total = 16.200
Turn
4P-4P-PF-Con * 4P-4P-F-Con * 4P-2P-T-Con = 115.200
4P-4P-PF-Con * 4P-3P-F-Con * 3P-2P-T-Con = 115.200
4P-3P-PF-Con * 3P-3P-F-Con * 3P-2P-T-Con = 108.000
4P-3P-PF-Con * 3P-2P-F-Con * 2P-2P-T-Con = 64.800
4P-2P-PF-Con * 2P-2P-F-Con * 2P-2P-T-Con = 29.160
4P-4P-PF-Con * 4P-2P-F-Con * 2P-2P-T-Con = 51.840
Total = 432.360

C Source Code für den Betting Tree Generator

C.1 GameStateNode

```
package bettingTree;

import java.util.LinkedList;
import java.util.Vector;

public class GameStateNode {
        public Vector<String> actions;
        public LinkedList<GameStateNode> children;
        public int current_player;
        public int first_player;
        public GameRules rules;
        public int[] bets;
        public boolean[] has_folded;

        public GameStateNode(GameRules rules) {
                this.actions = new Vector<String>();
                this.children = null;
                this.current_player = rules.firstPlayer;
                this.first_player = rules.firstPlayer;
                this.bets = rules.bets;
                this.has_folded = new boolean[rules.numberOfPlayers];
                this.rules = rules;

        }

        public GameStateNode(GameStateNode state) {
                this.actions = new Vector<String>(state.actions);
                this.current_player = state.current_player;
                this.first_player = state.first_player;
                this.bets = new int[state.bets.length];
                for (int i = 0; i < this.bets.length; ++i) {
                        this.bets[i] = state.bets[i];
                }
                this.has_folded = new boolean[state.has_folded.length];
                for (int i = 0; i < this.has_folded.length; ++i)
                        this.has_folded[i] = state.has_folded[i];
                this.rules = state.rules;
        }

        public static void currentPlayerRaises(GameStateNode state) {
                addAction(state, "r");
                state.bets[state.current_player] = getHighestBet(state) +
                   1;
```

```
        selectNextPlayer(state);
    }

    public static void currentPlayerCalls(GameStateNode state) {
        addAction(state, "c");
        state.bets[state.current_player] = getHighestBet(state);
        selectNextPlayer(state);
    }

    public static void currentPlayerFolds(GameStateNode state) {
        addAction(state, "f");
        state.has_folded[state.current_player] = true;
        selectNextPlayer(state);
    }

    public static boolean gameFinished(GameStateNode state) {
        if (state.actions.size() < state.rules.numberOfPlayers)
                return false;
        int highestBet = getHighestBet(state);
        for (int i = 0; i < state.bets.length; ++i) {
                if (state.bets[i] < highestBet && state.has_folded[
                   i] == false)
                        return false;
        }
        return true;
    }

    public static int getHighestBet(GameStateNode state) {
        int highestBet = 0;
        for (int i = 0; i < state.bets.length; ++i) {
                if (state.bets[i] > highestBet)
                        highestBet = state.bets[i];
        }
        return highestBet;
    }

    private static void addAction(GameStateNode state, String action) {
        state.actions.add(action +"(P" + state.current_player + ")
           ");
    }

    private static void selectNextPlayer(GameStateNode state) {
        int nextPlayer = state.current_player;
        do {
                if (nextPlayer + 1 < state.bets.length)
                        nextPlayer += 1;
                else
                        nextPlayer = 0;
        } while (state.has_folded[nextPlayer] && nextPlayer !=
           state.current_player);
```

```
        state.current_player = nextPlayer;
    }

    public static LinkedList<GameStateNode> getChildNodes(GameStateNode
        currentState) {
        LinkedList<GameStateNode> childNodes = new LinkedList<
           GameStateNode>();
        GameStateNode newState;

        if (GameStateNode.gameFinished(currentState))
                return null;

        // check or call is always an option
        newState = new GameStateNode(currentState);
        GameStateNode.currentPlayerCalls(newState);
        childNodes.add(newState);

        // check if fold is reasonable
        newState = new GameStateNode(currentState);
        if (GameStateNode.getHighestBet(currentState) >
           currentState.bets[currentState.current_player]) {
                GameStateNode.currentPlayerFolds(newState);
                childNodes.add(newState);
        }

        // check if raise cap is not reached yet
        newState = new GameStateNode(currentState);
        if (GameStateNode.getHighestBet(currentState) <
           currentState.rules.bettingCap) {
                GameStateNode.currentPlayerRaises(newState);
                childNodes.add(newState);
        }
        return childNodes;
    }

    public void generateChildNodes() {
        children = getChildNodes(this);
        if (children != null) {
                for (GameStateNode child : children)
                        child.generateChildNodes();
        }
    }

    public static void printNodeHistory(GameStateNode state) {
        System.out.println(state.actions);
    }

    public static int countSequences(GameStateNode state) {
        int ret = 0;
        if (!GameStateNode.gameFinished(state)) {
```

```
                ret = 1;
                if (state.rules.printActions)
                        GameStateNode.printNodeHistory(state);
                for (GameStateNode child : state.children)
                        ret = ret + GameStateNode.countSequences(
                           child);
        }
        return ret;
}

public static int countnPlayerSequences(GameStateNode state, int n)
    {
        int ret = 0;
        if (!GameStateNode.gameFinished(state)) {
                if (GameStateNode.nPlayersRemaining(state, n)) {
                        ret = ret + 1;
                        if (state.rules.printActions)
                                GameStateNode.printNodeHistory(
                                   state);
                }
                for (GameStateNode child : state.children)
                        ret = ret + GameStateNode.
                           countnPlayerSequences(child, n);
        }
        return ret;
}

public static int countContinuingNodes(GameStateNode state) {
        int ret = 0;
        if (gameFinished(state) && getPlayersRemaining(state) > 1)
           {
                if (state.rules.printActions)
                        GameStateNode.printNodeHistory(state);
                return 1;
        } else {
                if (!gameFinished(state))
                        for (GameStateNode child : state.children)
                                ret = ret + GameStateNode.
                                   countContinuingNodes(child);
        }
        return ret;
}

public static int countnPlayerContinuingNodes(GameStateNode state,
   int n) {
        int ret = 0;
        if (nPlayersRemaining(state, n) && gameFinished(state)) {
                if (state.rules.printActions)
                        GameStateNode.printNodeHistory(state);
                return 1;
```

```
        } else {
                if (!gameFinished(state))
                        for (GameStateNode child : state.children)
                                ret = ret + GameStateNode.
                                   countnPlayerContinuingNodes(
                                   child, n);
        }
        return ret;
}

public static int countnPlayerNodes(GameStateNode state, int n) {
        int ret = 0;
        if (GameStateNode.nPlayersRemaining(state, n)) {
                ret = ret + 1;
                if (state.rules.printActions)
                        GameStateNode.printNodeHistory(state);
        }
        if (!GameStateNode.gameFinished(state)) {
                for (GameStateNode child : state.children)
                        ret = ret + GameStateNode.countnPlayerNodes
                           (child, n);
        }
        return ret;
}

public static int countTerminalNodes(GameStateNode state) {
        int ret = 0;
        if (nPlayersRemaining(state, 1) && gameFinished(state)) {
                if (state.rules.printActions)
                        GameStateNode.printNodeHistory(state);
                return 1;
        } else {
                if (!gameFinished(state))
                        for (GameStateNode child : state.children)
                                ret = ret + GameStateNode.
                                   countTerminalNodes(child);
        }
        return ret;
}

public static int countInfosetActions(GameStateNode state) {
        int ret = 0;
        if (!gameFinished(state)) {
                for (GameStateNode child : state.children) {
                        ret = ret + 1 + countInfosetActions(child);
                        if (state.rules.printActions)
                                printNodeHistory(child);
                }
        }
        return ret;
```

```
}

public static int countnPlayerInfosetActions(GameStateNode state,
   int n) {
        int ret = 0;
        if (!gameFinished(state)) {
                for (GameStateNode child : state.children) {
                        if (nPlayersRemaining(state, n)){
                                ret = ret + 1 +
                                   countnPlayerInfosetActions(child
                                   , n);
                        if (state.rules.printActions)
                                printNodeHistory(child);
                        }
                        else
                                ret = ret +
                                   countnPlayerInfosetActions(child
                                   , n);
                }
        }
        return ret;
}

public static int getAllNodes(GameStateNode state) {
        int ret = 1;

        if (!GameStateNode.gameFinished(state)) {
                for (GameStateNode child : state.children)
                        ret = ret + GameStateNode.getAllNodes(child
                           );
        }
        return ret;
}

private static boolean nPlayersRemaining(GameStateNode state, int
   numberOfPlayers) {
        int numPlayersRemaining = 0;
        for (int i = 0; i < state.has_folded.length; ++i) {
                if (!state.has_folded[i])
                        numPlayersRemaining = numPlayersRemaining +
                            1;
        }
        if (numPlayersRemaining == numberOfPlayers)
                return true;
        else
                return false;
}

private static int getPlayersRemaining(GameStateNode state) {
        int numPlayersRemaining = 0;
```

```
                for (int i = 0; i < state.has_folded.length; ++i) {
                        if (!state.has_folded[i])
                                numPlayersRemaining = numPlayersRemaining +
                                    1;
                }
                return numPlayersRemaining;
        }
}
```

C.2 GameRules

```
package bettingTree;

public class GameRules {
        protected int numberOfPlayers;
        protected int firstPlayer;
        protected int firstToAct;
        protected int bettingCap;
        protected int[] bets;
        boolean printActions;

        public GameRules(int numPlayers, int firstP, int firstA, int
           bettingC, int[] bets, boolean print) {
                numberOfPlayers = numPlayers;
                firstPlayer = firstP;
                firstToAct = firstA;
                bettingCap = bettingC;
                this.bets = bets;
                printActions = print;
        }

        public static GameRules getnPlayerPreflop(int n, boolean print) {
                int[] bets = new int[n];
                for (int i = 0; i < n; ++i) {
                        if (i == 0)
                                bets[i] = 1;
                        else
                                bets[i] = 0;
                }
                return new GameRules(n, 0, 1, 4, bets, print);
        }

        public static GameRules getnPlayerFlopTurnRiver(int n, boolean
           print) {
                int[] bets = new int[n];
                return new GameRules(n, 0, 0, 4, bets, print);
        }
}
```