Opponent Modeling im Poker

Lars Meyer atomic_ant@freenet.de
Thomas Görge thgoerge@gmx.de
Technische Universität Darmstadt

Abstract

Eine starke Poker-KI zu entwickeln ist eine anspruchsvolle Aufgabe. Diese muss in der Lage sein, jeden ihrer Gegner einzuschätzen und ihr Spiel bestmöglich an diesen anzupassen um die maximale Gewinnrate zu erzielen. In der Einführung dieses Papers wird zunächst erläutert, warum das Modellieren des Gegners im Poker von großer Bedeutung ist. Danach werden die Ansätze beschrieben, die von den Programmen Loki bzw. Poki und Vexbot verwendet werden, um ihre Gegner zu beurteilen. Hierbei zeigen wir auch typische Probleme und Lösungsmöglichkeiten für diese auf. Abschließend werden die Ergebnisse von Experimenten vorgestellt, in denen die genannten Programme gegen andere Programme oder menschliche Gegner gespielt haben.

1. *Einführung*

Für das Spiel Schach wurden in der Vergangenheit bereits sehr starke Programme entwickelt, die die besten menschlichen Spieler schlagen können. Das Gleiche für ein Programm, das Poker spielt, zu bewerkstelligen, ist das Ziel vieler wissenschaftlicher Studien. Poker unterscheidet sich jedoch in einem wichtigen Punkt von Schach. Dem Spieler stehen nicht alle für seine Entscheidung relevanten Informationen zur Verfügung, denn man kennt das Blatt des Gegners nicht. Die optimale Entscheidung zu treffen bedeutet im Poker genau die Aktion durchzuführen, die man auch ausführen würde, wenn man die Hände der anderen Spieler kennt. Um also eine möglichst optimale Entscheidung zu treffen, muss man abschätzen, welche Karten der Gegner gerade auf der Hand halten könnte. Um dies durchzuführen müssen die Aktionen des Gegners beobachtet werden, um ein Modell seiner Spielweise zu erstellen. Das eigene Spiel hängt also sehr stark von den Strategien der anderen Spieler ab. Dies stellt einen Gegensatz zu Schach dar, denn hier ist die Spielweise des Gegners eher unbedeutend und es genügt den objektiv besten Zug auszuführen.

Da Poker eine mathematische Struktur zu Grunde liegt, existiert in der Theorie eine optimale Strategie, welche sich aus einem Nash Equilibrium ergibt. Es ist jedoch zur Zeit noch nicht möglich, eine solche Strategie mit Hilfe von Computern zu berechnen. Es gibt Ansätze, in denen eine Annäherung an diese optimale Strategie verwendet wird, um Texas Hold'em mit zwei Spielern zu spielen. Da dies jedoch nur eine Annäherung ist, können die Fehler in dieser durch starke Spieler entdeckt und ausgenutzt werden.

Selbst wenn eine optimale Strategie gefunden wird, reicht es nicht diese anzuwenden. In einem Nash Equilibrium hat keiner der Spieler den Anreiz seine Strategie zu wechseln, da dies nur zu einem schlechteren Ergebnis führen kann. Es wird also das minimale Ergebnis des Spielers maximiert, der eine solche Strategie anwendet. Die Equilibrium-Strategie für das Spiel Stein-Schere-Papier wäre zum Beispiel, rein zufällig eine der drei Möglichkeiten zu

wählen. Ein Spieler, der diese Strategie anwendet, kann auf lange Sicht niemals verlieren. Angenommen der Gegner wäre aber sehr schwach und würde immer nur die Möglichkeit Papier wählen. Gegen einen solchen Spieler würde man mit der Equilibrium-Strategie nicht gewinnen. Beobachtet man seinen Gegner jedoch, sieht man, dass er die ersten 10 mal nur Papier genommen hat und nimmt daher die nächsten Male immer Stein, um seine Schwäche auszunutzen.

Die Equilibrium-Strategie ist also eine defensive Strategie, die perfekt spielende Gegner annimmt. Dies ist aber beim Poker nicht der Fall und daher ist es einer der wichtigsten Aspekte, seinen Gegner zu beobachten und einzuschätzen um seine Schwächen schließlich auszunutzen. Eine Beobachtung könnte zum Beispiel die Häufigkeit sein, mit der ein Spieler vor dem Flop erhöht. Angenommen ein Spieler erhöht in 30% der Fälle vor dem Flop. Dann kann man davon ausgehen, dass er im Schnitt eine Hand hat, die zwischen den besten 15% und den besten 16% aller Hände liegt. Hält man selber eine Hand, die zu den 10 % der stärksten Hände gehört, kann man also nochmals erhöhen. Gegen einen Spieler der nur in 10 % der Fälle vor dem Flop erhöht, würde man die selbe Hand eher folden.

2. Einschätzen der eigenen Karten

Ein guter Poker Spieler muss die Situation, in der er sich befindet, sehr gut einschätzen können. Hierbei sollte er sich zunächst ein Bild über die Stärke sowie das Potential der eigenen Hand machen. Hiernach muss der Gegner betrachtet werden, um abhängig von dessen Spielweise eine möglichst optimale Aktion zu wählen. Im folgenden Abschnitt wird zunächst aufgezeigt, wie die Stärke sowie das Potential der eigenen Hand berechnet werden kann. Anschließend werden Verfahren beschrieben, um ein Modell des jeweiligen Gegners zu gewinnen und so das Spiel optimal auf diesen einzustellen. Hierbei werden die Ansätze der Bots Loki(Billings, Papp, Schaeffer, & Szafron, 1998)/Poki(Davidson, Billings, Schaeffer, & Szafron, 2000) sowie Vexbot(Davidson, Darse Billings, & Holte, 2004) verdeutlicht. Loki/Poki sind für die Poker Variante Fixed-Limit mit mehreren Mitspielern entwickelt worden, während Vexbot für das Heads-Up Spiel im Fixed-Limit entworfen wurde.

2.1 Einschätzen der eigenen Hand

Um die eigene Hand einzuschätzen müssen im wesentlichen zwei Eigenschaften dieser untersucht werden. Zum einen die momentane Stärke der Hand und zum anderen das Potential, dass die Hand sich noch verbessert.

2.1.1 Stärke der Hand

Mit der Stärke der Hand wird die Wahrscheinlichkeit bezeichnet, dass ein Spieler momentan die beste Hand hat. Die Stärke einer Hand muss in jeder Setzrunde neu bewertet werden. Angenommen man hat die Karten Ad [1] und Qc und der Flop wäre 3h-4c-Jh. Um die Stärke der eigenen Hand zu berechnen, kann man alle möglichen Hände, die der Gegner haben könnte durchgehen und zählen, wie viele davon besser, schlechter oder genauso gut sind wie die eigenen Karten. Der Gegner kann $\binom{47}{2}$= 1081 Kartenkombinationen auf der Hand halten. Hiervon sind 444 besser, 9 gleich gut und 628 schlechter als die eigene Hand.

1. Die Buchstaben d,h,s und c stehen für die Farbe der Karte (d = Karo, h = Herz, s = Pik, c = Kreuz).

Gegen zwei zufällige Karten ergibt sich also eine Chance von 58,5% für die eigene Hand besser zu sein. Diese Wahrscheinlichkeit bezieht sich auf das Spiel gegen einen Gegner. Um die Wahrscheinlichkeit gegen mehrere Gegner zu berechnen, kann man die gegen einen Spieler bestimmte Wahrscheinlichkeit mit der Anzahl der Gegner potenzieren. Gegen 5 Gegner würde sich also eine Handstärke von $0,585^5 = 0,069$ ergeben. Die Chance hier die momentan besten Karten auf der Hand zu halten wäre also gerade mal 6,9%.

2.1.2 Potential der Hand

Die momentane Stärke der Hand reicht nicht aus, um ihre Qualität zu beurteilen. Angenommen die eigenen Karten sind 7h-8h und der Flop wäre Js-6h-5h. Hier gibt es viele Karten, die noch auf dem Turn oder dem River kommen können und welche die eigene Hand erheblich verbessern (zur Straße oder zum Flush). Obwohl man am Flop wahrscheinlich hinten liegt, gewinnt man zu 65,1% gegen zwei zufällige Karten des Gegners. Daher berechnet man die Wahrscheinlichkeit, dass sich eine Hand, die momentan hinten liegt, noch zur besten Hand verbessert (P_{pot}). Mit Hilfe der zwei gesammelten Informationen (Handstärke und Potential der Hand) kann man die Effektive Handstärke (EHS) berechnen. Diese ist die Summe der Wahrscheinlichkeit, dass man momentan vorne liegt und der Wahrscheinlichkeit, dass sich die eigenen Karten noch zur stärksten Hand verbessern.

$$EHS = HS_n + (1 - HS_n) * P_{pot}$$

3. Einschätzen des Gegners

Für die Setzstrategie können jetzt die EHS und die Pot odds in Betracht gezogen werden. Jedoch ist die Annahme falsch, dass alle Karten, die der Gegner halten kann, gleich wahrscheinlich sind. Im obigen Beispiel (eigene Karten: Ad-Qc, Flop: 3h-4c-Jh) wurde beispielsweise gesagt, dass die Handstärke gegen einen Gegner bei 0,585 liegt. Tatsächlich wird man jedoch weitaus öfter als in 58,5% der Fälle vorne liegen, da die meisten Hände, die Ad-Qc hier schlagen, höchstwahrscheinlich vor dem Flop weggeworfen worden wären, da sie eigentlich sehr schwach sind. Das Ziel ist es also, dem Gegner eine Menge von möglichen Karten zuzuweisen, die er auf der Hand halten könnte, um dann seine Chancen zu berechnen. Um diese möglichst effizient bestimmen zu können, müssen die Informationen, die man während des Spiels gegen ihn gewonnen hat, in Betracht gezogen werden.

3.1 Der Ansatz in Loki

Loki weist jeder möglichen Kartenkombination eines Gegners ein Gewicht zu, welches beschreiben soll, wie wahrscheinlich es ist, dass er diese Karten auf der Hand hält. Hierbei wird jedem Gegner ein eigenes Set an Gewichten zugewiesen, da sich die Spieler sehr stark in Bezug auf ihre Strategie unterscheiden können. Desweiteren werden die Gewichte durch die Aktionen des Gegners beeinflusst. Wenn ein Spieler beispielsweise erhöht, gibt dies Aufschluss über die Stärke seiner Hand, und die Hände, die zu diesem Zeitpunkt sehr stark sind, müssen höher gewichtet werden.

Zum einen müssen also die initialen Gewichte für die Karten jedes Gegners bestimmt werden, und zum anderen müssen diese Gewichte im Laufe einer Hand aufgrund der Aktio-

nen des Gegners angepasst werden. Im Folgenden werden die Ansätze der Entwickler von Loki beschrieben, um diese Probleme zu lösen.

3.1.1 Berechnung der initialen Gewichte

Die Evaluierung der eigenen Karten vor dem Flop ist im Poker ein maßgebender Faktor, um erfolgreich zu spielen. Es gibt $\binom{52}{2} = 1326$ unterschiedliche Startkombinationen, die sich in 169 verschiedene Handtypen unterteilen lassen. Für jede dieser Handtypen wurden 1.000.000 Spiele gegen jeweils 9 zufällige Hände durchgeführt. Aus den Ergebnissen konnte dann die sogenannte *income rate* für jede Hand berechnet werden. Diese entspricht der durchschnittlichen Anzahl von Einsätzen, die mit dieser Hand gewonnen werden. Die höchste *income rate* hat beispielweise ein Paar Asse, während eine 7 und eine 2 in unterschiedlichen Farben die geringste *income rate* hat.

Mit Hilfe des Wissens über die *income rate* der Karten, sowie über die Beobachtung der Häufigkeit, mit der ein Gegner vor dem Flop wegwirft, mitgeht oder erhöht, kann eine sinnvolle Abschätzung der Gewichte der gegnerischen Hände getroffen werden.

Angenommen ein Gegner würde mit 30% aller Hände vor dem Flop mitgehen. Das bedeutet, dass die Hände, mit denen er mitgeht, im Durchschnitt eine *income rate* von 200 haben. Diesen Wert nennt man Median einer Hand und er wird mit μ bezeichnet. Desweiteren nimmt man eine bestimmte Varianz σ an. In diesem Beispiel soll sich durch die Varianz eine Schwankung der income rate der vom Gegner gespielten Hände um 100 ergeben. Allen Händen deren *income rate* größer als 200 + 100 = 300 ist wird das Gewicht von 1.0 zugeweisen, da sie sehr wahrscheinlich sind. Im Gegensatz dazu wird allen Händen deren *income rate* kleiner als 200 - 100 ist das Gewicht 0.01 zugewiesen, da es sehr unwahrscheinlich ist, dass der Gegner mit einer solch schlechten Starthand mitgeht. Die Gewichte der Hände mit einer *income rate* zwischen 100 und 300 werden linear interpoliert. Die Median-Hand erhält hierbei das Gewicht 0.5.

Es gibt viel Potential, diesen Ansatz zu verbessern, da spezielle Verhaltensweisen eines Spielers nicht berücksichtigt werden. Vielleicht erhöht ein Spieler öfter gegen Gegner, die von ihm als schlecht eingestuft werden. Erhöht er dann gegen einen starken Spieler, hat er höchstwahrscheinlich eine sehr starke Hand. Da das Programm aber nur die Durchschnittswerte betrachtet, wird es die Hand des Spielers schlechter einschätzen als sie wirklich ist.

3.1.2 Anpassen der Gewichte

Die Aktionen des Gegners, während eine Hand gespielt wird, führen dazu, dass die Gewichte seiner Hände angepasst werden. Diese werden unterteilt in die Aktion, die er durchführt (Fold, Check/Call, Bet/Raise), wie viele Bets er zahlen müsste um mitzugehen (0 Bets, 1 Bet oder > 1 Bet) und wann die Aktion ausgeführt wird (Pre-Flop, Flop, Turn, River). Aktionen können also in 36 verschiedene Kategorien unterteilt werden. Für jede Kategorie wird die Häufigkeit bestimmt, mit der der Gegner die entsprechende Aktion ausführt. Diese Werte werden dafür verwendet, um die Anpassung der Gewichte vorzunehmen.

Beispielsweise könnte man durch die Analyse dieser Häufigkeiten herausfinden, dass ein Gegner eine effektive Handstärke (EHS) von durchschnittlich 0.6 braucht, um am Flop nach einer Erhöhung mitzugehen. Unter Einbeziehung der Varianz kann die untere Grenze 0.4

Hand	Weight	EHS	Rwt	Nwt
Jd-4h	0.01	0.99	1.00	0.01
Ac-Jc	1.00	0.94	1.00	1.00
5s-5h	0.70	0.74	0.85	0.60
Qs-Ts	0.90	0.22	0.01	0.01

Table 1: Anpassung der Gewichte beispielhafter Hände nach dem Flop 3h-4c-Jh

und die obere Grenze 0.8 sein. Jetzt werden die Gewichte aller Hände, deren EHS kleiner als 0.4 ist, mit einem sehr kleinen Faktor (z.B. 0.01) multipliziert, da der Gegner mit solchen Händen wahrscheinlich nicht mitgehen würde. Die Gewichte aller Hände mit einer EHS $>$ 0.8 werden mit dem Faktor 1 multipliziert, bleiben also gleich. Die Faktoren für die Hände mit einer EHS zwischen 0.4 und 0.8 werden durch linerare Interpolation bestimmt. Die Anpassung der Gewichte wird für jede Setzrunde durchgeführt, so dass am Ende nur noch sehr wenige Hände ein hohes Gewicht haben.

In Tabelle 1 werden die Werte, die für die Anpassung der Gewichte wichtig sind, für einige interessante Beispielhände dargestellt. Die Werte stehen für einen Gegner, der vor dem Flop mitgegangen ist und jetzt am Flop (3h-4c-Jh) setzt. Weight bezeichnet das Gewicht, das der Hand vor dem Flop zugewiesen wurde, EHS die effektive Stärke der Hand, Rwt den Faktor, mit dem das Gewicht multipliziert wird und Nwt das neue Gewicht der Hand.

Die Hand Jd-4h wäre die stärkste Kartenkombination im Beispiel. Jedoch ist sie vor dem Flop schwach und erhält daher ein Gewicht von 0.01, wenn man davon ausgeht, dass der Spieler in 30% der Fälle vor dem Flop mitgeht. Da der EHS der Hand über 0.8 liegt beträgt der Faktor zur Anpassung der Gewichte 1, so dass das neue Gewicht bei 0.01 bleibt.

Eine starke Hand, die zudem sehr wahrscheinlich ist, ist Ac-Jc. Hier beträgt das Gewicht am Ende weiterhin 1.

Qs-Ts sind vor dem Flop relativ gute Karten, was sich in einem Gewicht von 0.90 ausdrückt. Jedoch wurde auf dem Flop nichts getroffen und daher beträgt der EHS der Hand am Flop nur 0.22. Aufgrund der genannten Beobachtungen gehen wir wieder davon aus, dass die Durchschnittsstärke einer Hand 0.6 betragen muss, damit der Gegner setzt und sich durch die Varianz eine Schwankung von $\pm$0.2 ergibt. Da die EHS von Qs-Ts kleiner als 0.4 ist, wird das Gewicht der Hand mit dem Faktor 0.01 multipliziert.

Die hier aufgezeigten Verfahren bieten zwar schon gute Ansätze, man muss jedoch beachten, dass es noch viel Raum für Verbesserungen gibt. Gegner ändern beispielsweise ihre Taktik und spielen am Anfang des Spiels sehr konservativ und tight um dieses Image im späteren Verlauf des Spiels auszunutzen. Desweiteren ist die Kategoriesierung der Situationen, für die die Häufigkeiten der Aktionen gezählt werden, sehr grob. Viele relevante Informationen, wie beispielsweise die eigene Position zu den Gegnern, werden außer Acht gelassen.

3.1.3 Weiterentwicklung von Loki durch Poki

Wie in dem obigen Abschnitt bereits erwähnt wurde, werden in Loki die Häufigkeiten der Aktionen des Gegners in unterschiedlichen Situationen gemessen. Es wurde eine sehr ein-

	FOLD	CALL	RAISE	%
FOLD	13.0	0.3	0.3	13.6
CALL	0.0	58.4	3.3	61.8
RAISE	0.0	10.5	14.1	24.7
%	13.0	69.3	17.7	85.6

Table 2: Genauigkeit der Vorhersagen des Neuralen Netzes

fache Kategorisierung vorgenommen, welche 12 verschiedene Typen von Situationen unterscheidet, in denen ein Spieler jeweils wegwerfen, mitgehen oder erhöhen kann. Die Kategorien ergaben sich aus der Setzrunde (Pre-Flop, Flop, Turn, River) und der Anzahl der Bets, die eingesetzt werden müssen, um mitzugehen (0,1,2 oder mehr). Diese Kategorisierung ist sehr vereinfacht und lässt viele relevante Details außer Acht. Es ist beispielsweise sehr wichtig, wie viele Spieler sich zu dem Zeitpunkt noch im Spiel befinden. Desweiteren ist etwa das Setzen eines Spielers niedriger zu bewerten, wenn dieser in letzter Position sitzt und vorher alle Gegner gecheckt haben.

Jedoch kann nicht jeder Faktor für die Kategorisierung verwendet werden, da diese dann viel zu komplex wird. Um festzustellen welche Faktoren am relevantesten sind, wurde eine frühere Studie betrachtet, in der mit Hilfe eines *artificial neural networks* (ANN) die nächste Aktion des Gegners vorausgesagt werden sollte. Als Trainingsdaten standen mehrere hundert Hände dieses Spielers zur Verfügung. Die Eingangsknoten des ANN waren Parameter, die den Kontext der Situation beschreiben (beispielsweise ob die letzte Aktion ein Call oder ein Raise war oder ob ein Ass auf dem Board liegt). Die Ausgangsknoten gaben die Vorhersage für die Aktion des Gegners an (Fold,Bet,Raise). Tabelle 2 zeigt die Genauigkeit der Vorhersagen eines typischen ANN. Die Spalten geben hierbei an, welche Aktion vorausgesagt wurde, während die Zeilen darstellen was der Gegner wirklich gemacht hat. In diesem Fall hat der Gegner zum Beispiel in 13.6% aller Fälle gefoldet. In 13.0% aller Fälle wurde ein Fold korrekt vorhergesagt, während ein Call und ein Raise jeweils zu 0.3 % der Fälle vorausgesagt wurde, wenn der Spieler in Wirklichkeit gefoldet hat. Insgesamt konnten hier zu 85.6 % die Aktion des Spielers richtig vorhergesagt werden.

Ein solches ANN kann nicht während des Spiels eingesetzt werden, da die Berechnung zu lange dauert. Jedoch erhielt man als Ergebnis dieser Studie eine Gewichtung für die Relevanz der Eingangsparameter. Zwei Parameter stellten sich hierbei als besonders wichtig heraus. Hierbei handelte es sich um die vorangegangene Aktion des Gegners sowie um die Anzahl der Bets, die der Gegner vorher aufbringen musste um mitzugehen. Die Kategorisierung der Situationen wurde um diese Parameter erweitert.

Durch diese Erweiterungen konnte die Gewinnrate des Bots signifikant verbessert werden. Im Abschnitt **Experimente** werden wir genauer hierauf eingehen.

3.2 Vexbot

Im folgenden Abschnitt wird das Programm VEXBOT vorgestellt, dass mit Hilfe von Suchbäumen den zu erwartenden Gewinn bzw. Verlust der verschiedenen Handlungsmöglichkeiten berechnet, um eine Entscheidung zu treffen. Dabei werden nicht nur die eigene Handstärke und statischen Gewinnwahrscheinlichkeiten, sondern auch Erfahrungswerte über den Geg-

ner berücksichtigt, um eine bessere Gewinn- bzw. Verlustschätzung zu erreichen. Eine Neuerung von VEXBOT ist hier seine Fähigkeit trotz unvollständiger Informationen selbstständig gewinnmaximierende Konterstrategien zu entwickeln. VEXBOT wurde speziell für das Spiel gegen genau einen Gegner entwickelt und stellt neben dem Sieg gegen die vormals beste KI "PsOpti" auch eine echte Herausforderung für starke menschliche Spieler dar.

Die gesamten Berechnungen von VEXBOT beruhen dabei auf einer Baumdarstellung in der alle möglichen Spielverläufe von der aktuellen Situation bis zum Ende der Runde enthalten sind. Im Baum wird zwischen Knoten unterschieden bei denen das Programm selbst eine der drei möglichen Aktionen: "fold", "check/call", "bet/raise" auswählen muss, den sog. "Program decision nodes" und solchen bei denen der Kontrahent eine Entscheidung zu treffen hat: "Opponent decision nodes". Blattknoten sog. "Leaf nodes" repräsentieren das Ende einer Runde, das entweder durch den "fold" eines Spielers oder durch den "showdown" zu stande kommt. Abschließend stellen "Chance Nodes" die zufällige Komponente von Poker nämlich das Aufdecken der öffentlichen Karten dar.

Für jeden dieser Knoten lässt sich ein erwarteter Gewinn berechnen, der als Entscheidungskriterium dient. Beim "Chance Node" ergibt sich dieser einfach aus der Wahrscheinlichkeit, dass eine bestimmte Karte C_i aufgedeckt wird: $Pr(C_i)$ und dem erwarteten Gewinn des entsprechenden Teilbaums: $EG(C_i)$

$$EG(C) = \sum_{1 \leq i \leq n} Pr(C_i) \times EG(C_i)$$

Auch beim "Opponent decision node" lässt sich der erwartete Gewinn als gewichtete Summe je nach getroffener Entscheidung O_i des Kontrahenten berechnen:

$$EG(O) = \sum_{i \in \{f,c,r\}} \sum Pr(O_i) \times EG(O_i)$$

Bei "Program decision nodes" kann der erwartete Gewinn unter Verwendung der sog. "Miximax" Strategie wie folgend angegeben werden:

$$EG(U) = \max(EG(U_{fold}), EG(U_{check/call}), EG(U_{bet/raise}))$$

Hier ist jedoch auch eine sog. "Miximix" Strategie mit einer beliebigen Gewichtung ähnlich wie bei "Chance Nodes" und "Opponent Nodes" denkbar, die zu einer zufälligen Wahl der Aktion führt.

Zu guterletzt lässt sich noch der erwartete Gewinn der Blattknoten in Abhängigkeit von der Gewinnwahrscheinlichkeit, der Potgröße und der in den Pot selbst eingezahlten Geldmenge beschreiben:

$$EG(L) = (P_{win} \times L_{\$pot}) - L_{\$cost}$$

Da an jedem inneren Entscheidungsknoten drei Handlungsmöglichkeiten bestehen und in jeder der 4 Spielphasen: "Preflop", "Flop", "Turn" und "River" 4 mal der Einsatz erhöht werden darf, kommt es leicht zu Bäumen mit mehreren Millionen Knoten. Dies macht eine effiziente Datenstruktur und das Caching von Zwischenergebnissen notwendig, um die erwarteten Gewinne trotzdem noch in annehmbarer Zeit zu berechnen.

Die besondere Herausforderung bzgl. des Opponent Modeling ist zum einen die Wahrscheinlichkeit mit der der Gegner bei einem "Opponent decision node" eine bestimmte Aktion

ausführt möglichst genau zu berechnen und zum anderen den erwarteten Gewinn eines Blattknotens zu bestimmen. Wird die Runde durch das folden eines Teilnehmers beendet, so ist die Berechnung relativ trivial, da die Gewinnwahrscheinlichkeit entweder 0% oder 100% beträgt. Sobald es jedoch zu einem showdown kommt, muss hier mit Hilfe des Opponent Modelings eine möglichst zutreffende Schätzung erstellt werden. VEXBOT versucht dies bestmöglich zu gewährleisten indem es das Setzverhalten die sog. "bet-sequence" zusammen mit der Handstärke des Kontrahenten von jedem gesehenen Showdown speichert. In einem späteren Spiel wird dann geprüft, ob bereits eine ähnliche "bet-sequence" beobachtet wurde und wie stark seine Hand in diesen Fällen war. Die Handstärke wird bei VEXBOT in Intervalle zwischen 0 und 1 unterteilt und deren Auftrittshäufigkeit bei einer bestimmten "bet-sequence" in einem Histogramm gespeichert. Sollte von einem Gegner bei einer bestimmten "bet-sequence" 7 mal eine extrem starke und nur einmal eine sehr schwache Hand beobachtet werden so käme bei 10 Intervallen folgendes Histogramm zu stande: [0 1 0 0 0 0 0 0 7 0]. Man erhält somit die Information, dass er bei dieser "bet-sequence" in 87,5% der Fälle tatsächlich sehr gute Karten einer Stärke vom 0,8-0,9 und nur in 12,5% der Fälle mit einer Handstärke von nur 0,1-0,2 geblufft hat.

In dem eben genannten Verfahren werden allerdings nur Beobachtungen berücksichtigt bei denen es zu einem Showdown kommt, damit man die Handstärke des Kontrahenten auch erfährt. Sehr häufig ist aber gerade dies nicht der Fall, da oft einer der beiden Spieler die Runde vorzeitig durch folden beendet. Auch diese Situationen werden von VEXBOT berücksichtigt, da man trotz unbekannter Handstärke des Gegners die Häufigkeiten für ein bestimmtes Verhalten bzw. bestimmte "bet-sequences" beobachten kann. Leicht kann man sich verdeutlichen, dass die Information, dass ein Spieler immer foldet sobald sein Kontrahent raised sehr wertvoll ist, obwohl man nie seine Handstärke erfährt. Diese Häufigkeiten bieten somit einen Anhaltspunkt dafür, wann und wie der Kontrahent entscheidet.

3.2.1 Anwendungsbeispiel

Im Folgenden wird versucht die Vorgehensweise von VEXBOT durch ein kurzes Beispiel etwas anschaulicher darzustellen und seine Funktionen Schritt für Schritt aus dessen Sichtweise zu erklären. In der dargestellten Situation befinden wir uns in der letzten Setzrunde direkt nachdem der River aufgedeckt wurde. Der Pot entspricht zu Beginn der Runde der vierfachen Größe des Big Blinds(BB). Nach einem bet unsererseits (Spieler1) reagiert der Gegner (Spieler2) mit einem raise, wir befinden uns nun also an dem rot markierten Knoten in Figur1. Um zu entscheiden wie wir nun reagieren sollen werden die erwarteten Gewinne der möglichen Aktionen berechnet: **EG(fold)**, **EG(call)** und **EG(raise)**.

EG(fold) : Ein fold unsererseits führt offensichtlich zum Verlust von allem was wir bisher gesetzt haben, also von 2 BB vor dem River und 2 BB durch unseren bet zu einem EG(fold)= -4 BB.

EG(call) : Um den EG(call) berechnen zu können benötigen wir nun allerdings Informationen wie die Handstärke des Gegners, mit der er typischerweise ein solches Verhalten gezeigt hat und natürlich die Stärke der eigenen Hand. Angenommen durch vorherige Spiele liegt uns bereits folgendes Histogramm vor, das Beobachtungen über exakt den selben Spielverlauf bzw. "bet-sequence" enthält: [1 1 0 0 0 0 0 0 4 4

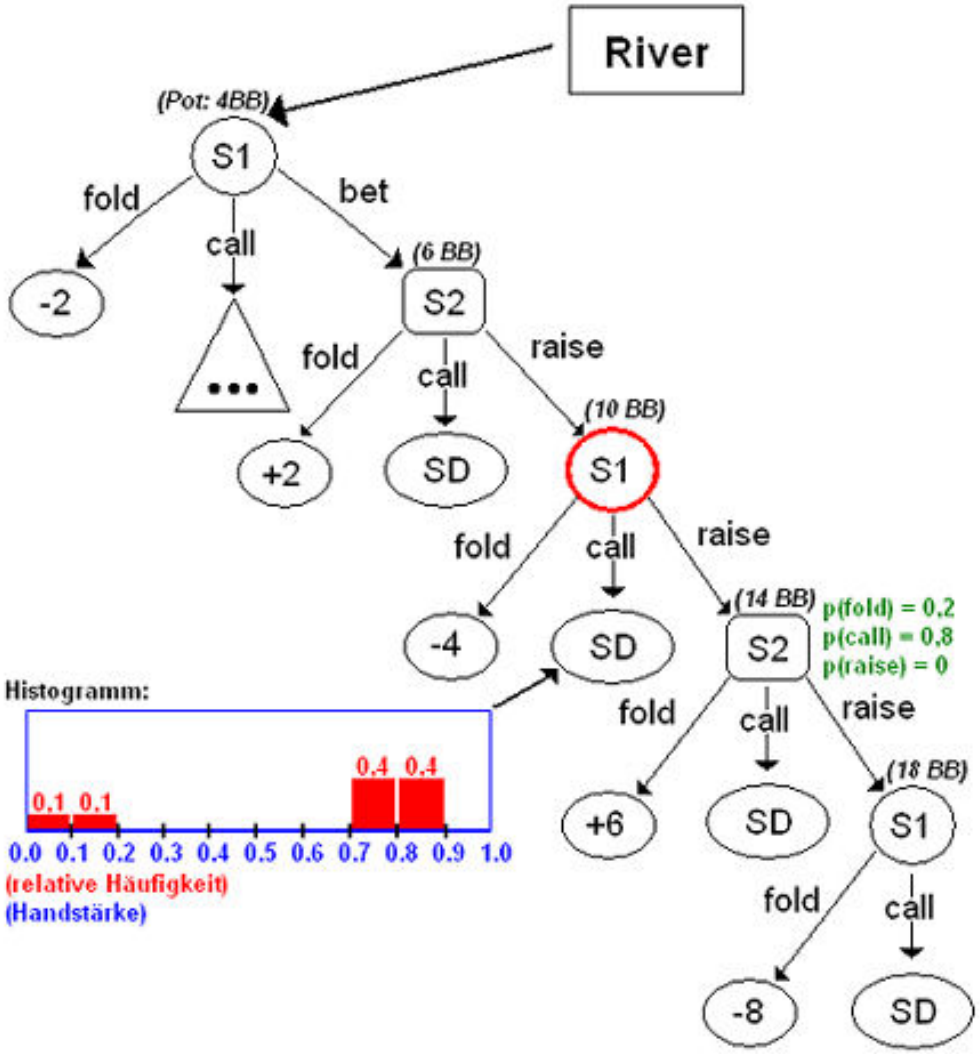

Figure 1: beispielhafter Ausschnitt des Suchbaums

0]. Dies bedeutet, dass der Gegner in der Vergangenheit in 20% der vergleichbaren Fälle nur eine Handstärke zwischen 0 und 0,2 hatte(siehe [1 1 ...]) und in 80% der Fälle eine Handstärke von 0,7 bis 0,9 (siehe [... 4 4 0]). Entsprechen unsere eigenen Karten nun zum Beispiel einer Handstärke zwischen 0,2 und 0,7 so könnten wir besten Wissens nur gegen einen Bluff gewinnen, der bisher in 20% der Fälle auftrat. Folglich liegt unsere Gewinnwahrscheinlichkeit bei einem call Pr(win—call) = 20%. Da sich der Pot durch unseren call nun auf 12 BB ausgeweitet hat, von denen 6 BB vom Gegner stammen, würden wir durch einen Sieg also genau 6 BB gewinnen: $Pr(win|call) * 6BB = 0,2 * 6BB = 1,2BB$. Nicht zu vergessen ist allerdings die Tatsache, dass wir in 80% der Fälle auch die 6 von uns gesetzten Big Blinds

HS	Pr(win/call)	EG(call)	Pr(win/raise,call)	EG(raise)	beste Aktion
0,70	0,2	-3,6	0,0	-5,2	call
0,75	0,4	-1,2	0,25	-2,0	call
0,80	0,6	+1,2	0,5	+1,2	call oder raise
0,85	0,8	+3,6	0,75	+4,4	raise
0,90	1,0	+6,0	1,0	+7,6	raise

Table 3: Erwartete Gewinne (EG) von call und raise in Abhängigkeit der eigenen Handstärke (HS)

verlieren: $Pr(loose|call) * -6BB = 0,8 * -6BB = -4,8BB$. Dies führt insgesamt zu einem erwarteten Gewinn von: $EG(call) = 1,2BB - 4,8BB = -3,6BB$ und stellt im Gegensatz zum Folden eine Verbesserung um 0,4 BB dar. Sollte unsere eigene Handstärke allerdings nur 0,1 betragen ergibt sich unser erwarteter Gewinn zu $EG(call) = -4,8BB$ und es wäre besser für uns zu folden. Eine Übersicht über die erwarteten Gewinne im Falle eines Calls oder Raises in Abhängigkeit der eigenen Handstärke sind in Tabelle1 dargestellt.

EG(raise) : Um den erwarteten Gewinn eines re-raises unsererseits zu berechnen, müssen wir zunächst die erwarteten Gewinne der sich daraus ergebenen Möglichkeiten ermitteln und diese dann entsprechend ihrer Eintrittswahrscheinlichkeit gewichten. Diese Wahrscheinlichkeiten werden hier anhand der bisher beobachteten Häufigkeiten für bestimmte Aktionen unter vergleichbaren Umständen gewonnen. Der Einfachheithalber setzen wir die Wahrscheinlichkeit eines re-raises durch den Gegner auf 0% wodurch die Fälle c) und d) im weiteren Verlauf vernachlässigt werden können. Fall a) tritt zu 20% und b) folglich zu 80% auf.

Grundsätzlich kann es nach unserem re-raise zu 4 verschiedenen Szenarien kommen:

a) Der Gegner steigt aus: fold. In diesem Fall kommt es zu keinem Showdown, da der Gegner die Runde vorzeitig beendet. Zusammen mit einer beobachteten Wahrscheinlichkeit von 20% für diese Aktion berechnet sich der gewichtete, erwartete Gewinn also zu $0,2 * EG(raise, fold) = 0,2 * 6BB = 1,2BB$ und ist unabhängig von unserer Handstärke.

b) Der Gegner geht mit: call. Hier muss wie schon bei EG(call) beschrieben mit Hilfe der Handstärken eine Gewinnwahrscheinlichkeit und damit nun der erwartete Gewinn berechnet werden. Da der Pot hier schon auf 16 BB gewachsen ist beträgt er: $EG(raise, call) = 16 * Pr(win|raise, call) - 8$, also zum Beispiel: $EG(raise, call) = 16 * 0,0 - 8 = -8BB$ bei einer Gewinnwahrscheinlichkeit von 0%. Wir gehen also davon aus, dass er nur mit einer sehr starken Hand callen wird und wir hier keine Aussicht auf einen Sieg haben. Zusammen mit der Eintrittswahrscheinlichkeit dieses Falles von 80% ergibt sich der Anteil von b) an EG(raise) mit $0,8 * -8BB = -6,4BB$.

c) Der Gegner erhöht erneut, woraufhin wir folden: re-raise, fold.
$\Rightarrow 0 * EG(raise, raise, fold) = 0BB$

d) Der Gegner re-raised uns woraufhin wir callen: re-raise, call.
$\Rightarrow 0 * EG(raise, raise, call) = 0BB$

Mit Hilfe der einzelnen Anteile von EG(raise,fold), EG(raise,call), EG(raise,raise,fold) und EG(raise,raise,call) ergibt sich EG(raise) also als:

$$EG(raise) = 0,2{*}EG(raise, fold){+}0,8{*}EG(raise, call){+}0{*}EG(r, r, f){+}0{*}EG(r, r, c)$$

$$EG(raise) = 1,2BB + -6,4BB + 0BB + 0BB$$

$$EG(raise) = -5,2BB$$

Hier kann man ablesen, dass sich VEXBOT unter Verwendung der Miximax-Strategie bei einer eigenen Handstärke von 0,7 für einen call entscheidet, da hier mit EG(call) = -3,6BB der größte Gewinn zu erwarten ist. Sollte auf lange Sicht die Verwendung des jeweils größten Wertes zu vorhersehbaren Handlungen führen, ist hier allerdings auch eine Miximix-Strategie denkbar. Diese wählt zufällig eine Aktion aus, wobei zwar Aktionen mit größerem erwartetem Gewinn auch wahrscheinlicher durchgeführt werden, es einem Gegner aber trotzdem erschwert eine genaue Vorhersage über das eigene Verhalten zu treffen.

3.2.2 Probleme

Das Opponent Modeling beim Pokern ist eine große Herausforderung, die einige grundsätzliche Probleme mit sich bringt.

1. Das Modell des Kontrahenten muss hier sehr schnell erstellt werden, da ein durchschnittlich langes Match beim Pokern gegen menschliche Spieler meist aus weniger als 100 Runden besteht. Hier ist das Ziel möglichst alle zur Verfügung stehenden Informationen zu nutzen, um so den Lernprozess zu beschleunigen.

2. Erfahrene Pokerspieler wechseln häufig ihr Spielverhalten, um möglichst unberechenbar zu bleiben. Dies führt dazu, dass aus Vergangenheitsdaten nicht mehr auf sein aktuelles Verhalten geschlossen werden kann und der Lernprozess von vorne beginnen muss. Ein solcher Strategiewechsel des Gegners muss auch desshalb möglichst schnell erkannt werden da das bisher für ihn aufgestellte Modell zu Fehlentscheidungen führen würde und so viel Schaden anrichten kann.

3. Eine weitere Schwierigkeit mit der man beim Erstellen einer guten Poker KI konfrontiert ist, stellen die oft nur unvollständigen Informationen dar. In vielen Spielsituationen kommt es dazu, dass Spieler folden und so ihre Karten nicht aufgedeckt werden. Hier kann man nicht wissen ob sie wegen einer schwachen Hand kaum eine andere Möglichkeit hatten oder einen Fehler gemacht haben indem sie ein stärkeres Blatt weglegten. Auch der Fall, dass der Spieler ein relativ gutes Blatt weglegt, da er einen Kontrahenten anhand seiner Beobachtungen als stärker einschätzt, bleibt der KI verborgen.

Bei VEXBOT ist das 1. Problem besonders schwerwiegend, da hier Informationen immer in Verbindung mit genau einer “bet-sequence” gespeichert werden. Wie oben bereits beschrieben gibt es sehr viele Variationsmöglichkeiten bzgl. der “bet-sequences” was

leicht zu Suchbäumen mit vielen Millionen Knoten führt. Das Problem daran ist, dass somit jede dieser vielen "bet-sequences" mindestens einmal beobachtet werden muss, um für jeden Fall auch eine Prognose aufstellen zu können. Dies ist natürlich unter normalen Umständen kaum gewährleistet, da hier die Anzahl der Runden eines Spieles weit geringer ist als die Anzahl der unterschiedlichen "bet-sequences". Um diesem grundsätzlichen Problem zu begegnen hat man ein Abstraktionssystem entwickelt, dass je nach Umfang der zur Verfügung stehenden Informationen eine geeignete Abstraktionsschicht auswählt. Die Abstraktion bezieht sich dabei auf die Unterscheidung zwischen verschiedenen "bet-sequences" und weist solche mit gemeinsamen Eigenschaften einer gemeinsamen Gruppe zu. Dies führt dazu, dass nicht jede einzelne "bet-sequence" sondern nur jede Gruppe von "bet-sequences" beobachtet werden muss, um eine Prognose zu liefern. Je weniger Informationen verfügbar sind desto stärker muss abstrahiert also desto weniger Gruppen können unterschieden werden. Eine Gruppe, die zum Beispiel alle "bet-sequences" mit gleicher Anzahl an bets und raises enthält ohne zwischen der genauen Abfolge zu unterscheiden, enthält logischerweise viel mehr Beobachtungen als dies mit Berücksichtigung der Reihenfolge der Fall ist. Sie gibt somit zwar keine genaue Auskunft über sein Verhalten in einer bestimmten, vielleicht bisher noch nie beobachteten Situation, kann aber durch die Verfügbarkeit von ähnlichen Situationen wenigstens eine Schätzung abgeben an der man sich orientieren kann.

Hier kommt es also zu einem Trade-Off zwischen der mit größerer Abstraktion abnehmenden Relevanz der Gruppe für den aktuellen Einzelfall und der zunehmenden Anzahl der in der Gruppe enthaltenen Beobachtungen, die durch eine größere Datenbasis eher Rückschlüsse auf sein prinzipielles Verhalten zulassen. Bei Tests hat sich herausgestellt, dass es oft vorteilhafter ist eine stärker abstrahierende Unterteilung mit vielen Daten pro Gruppe zu verwenden, da die Prognose andernfalls zu stark von einzelnen, nicht repräsentativen Entscheidungen beeinflusst wird. Ein intuitiver Ansatz wäre also die Abstraktionsebene mit dem größten Informationsgehalt zu ermitteln und als Grundlage für das Opponent Modelling zu verwenden. Dies wird bei VEXBOT allerdings in einer leicht abgewandelten Form umgesetzt, indem vor jeder Entscheidung nicht nur eine, sondern alle Abstraktionsschichten, nach ihrem jeweiligen Informationsgehalt gewichtet, berücksichtigt werden.

Nehmen wir beispielsweise an, dass für eine aktuelle Showdown-Situation genau 5 Beobachtungen vorliegen, die eine identische "bet-sequence" besitzen. Diese sehr konkrete Information der wir ein Abstraktionslevel (A0 $\Rightarrow$ gar keine Abstraktion) zuweisen, würde von VEXBOT zum Beispiel für $m = 0,95$ mit $(1 - m^5) = 0,23$ des Gesamtgewichts berücksichtigt werden. Unter der Annahme, dass auf dem nächst höheren Abstraktionslevel (A1) 20 Beobachtungen vorhanden sind, die sich zum Beispiel in der Anzahl der bets und raises der jeweiligen Spieler gleichen, würde die hier gewonnene Information mit $(1 - m^{20}) = 0,64$ des noch verbliebenen Gewichts, also mit $(1 - 0,23) * 0,64 \approx 50\%$ des Gesamtgewichts berücksichtigt werden. Die nächst höhere Abstraktion (A3) enthält vielleicht 75 Beobachtungen und würde also mit $(1 - m^{75}) = 0,98$ beinahe das gesamte restliche Gewicht ca. 27% erhalten. Dies lässt sich so lange weiterführen, bis allen vorgesehenen Abstraktionsschichten ein Gewicht zugeordnet wurde, das deren Relevanz für die aktuelle Situation möglichst gut widerspiegelt.

Das zweitgenannte Problem, dass gute Spieler oft ihre Vorgehensweise ändern, berücksichtigt VEXBOT indem es neuere Daten stärker gewichtet und ältere Daten entsprechend einer

Methode exponentieller Glättung immer mehr vernachlässigt. Die zu Grunde liegende Funktion hängt dabei von einem Parameter h ab, der die genaue Gewichtung festlegt. Bei h = 0.95 würde zum Beispiel die neueste Beobachtung mit 5% und die letzten 1/(1-h) = 20 Beobachtungen mit insgesamt (1-1/e) = 63% des Gesamtgewichts in die Berechnung eingehen.

4. Experimente

In den folgenden Abschnitten werden die Testergebnisse der hier vorgestellten Bots beschrieben. Dabei unterscheiden wir die durchgeführten Experimente je nach Anzahl der Spielteilnehmer.

4.1 Mehrspieler-Matches

Poki wurde auf einem Online Poker Server im Internet Relay Chat (IRC) getestet. Dort spielen menschliche Spieler in verschiedenen Channels gegeneinander. Obwohl nicht um echtes Geld gespielt wird, spielen die meisten Spieler ernsthaft und somit handelt es sich hierbei um eine gute Testumgebung für Poki. Es ist allen Spielern erlaubt an den Einsteiger-Spielen teilzunehmen (#holdem1). Ein Spieler, der genug virtuelles Geld verdient hat, darf dann an Spielen mit fortgeschrittenen Gegnern teilnehmen (#holdem2).

Ein Ziel war es zu testen, ob sich die Gewinnrate durch das verbesserte System, mit dem Poki die Gegner modelliert, signifikant verbessert. Daher wurde Poki sowohl mit dem alten System, wie es von Loki verwendet wurde, als auch mit dem verbesserten System in den Einsteiger- und den Fortgeschrittenen-Spielen getestet. Zusätzlich wurde auch eine Version getestet, die kein Opponent Modeling verwendet. Um möglichst aussagekräftige Ergebnisse zu erzielen, sind jeweils über 20.000 Hände gespielt worden. Die Gewinnrate wird in Small Bets pro Hand (sb/h) gemessen.

Die Version ohne Opponent-Modeling erzielte in den Einsteiger-Spielen weder Gewinne noch Verluste. Das Programm mit dem alten System (Poki_s1) erzielte in den Einsteiger-Spielen eine Gewinnrate von 0.09 sb/h, während das Programm mit dem verbesserten Opponent Modeling (Poki_s2) sogar eine Gewinnrate von 0.22 sb/h erreichte. Die Gewinnrate eines professionellen Pokerspielers liegt in etwa zwischen 0.05 und 0.10 sb/h. Daher sind die Ergebnisse sehr gut, auch wenn die Gegner eines professionellen Pokerspielers weitaus stärker sind.

In den Fortgeschrittenen-Spielen erzielte Poki_s1 eine Gewinnrate von 0.09 sb/h und Poki_s2 eine von 0.08 sb/h. Allerdings kam es bei Poki_s2 scheinbar zu einer Anomalie zwischen der 5.000 und der 10.000 Hand, da der Gewinn hier fast komplett verloren wurde. Hierfür gibt es außer Pech keine sinnvolle Erklärung. Zwischen der 10.000 und der 20.000 Hand wurde wieder fast genauso viel virtuelles Geld verdient, wie Poki_s1 in dem Zeitraum von 20.000 Händen gewonnen hat. Abbildung 1 zeigt diese Ergebnisse. Falls diese Anomalie tatsächlich aufgrund von Pech entstanden ist, würde die Gewinnrate von Poki_s2 mindestens 0.12 sb/h betragen.

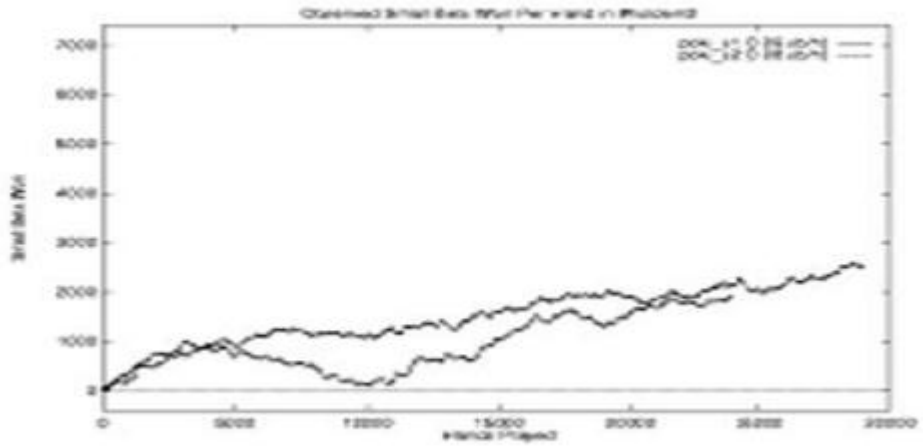

Figure 2: Ergebnisse von Poki_s1 und Poki_s2 in #holdem2

4.2 2-Spieler-Matches

Im Verlauf des 2-Spieler-Experiments spielten im Rahmen eines round-robin Tuniers alle teilnehmenden Pokerprogramme mindestens 40.000 Hände gegeneinander. Neben VEXBOT wurden zusätzlich noch folgende Programme im Experiment untersucht:

1. SPARBOT ist die öffentlich zugängliche Version von PsOPTI-4 und war bisher in Limit Hold'em mit 2 Spielern das stärkste bekannte Programm.

2. POKI ist die erweiterte Version von Loki und stellt ein Formelbasiertes Programm dar, das mit Hilfe von Opponent Modeling besser auf das Verhalten des Gegners reagieren kann. Es ist das stärkste bekannte Programm in der 10-Spieler-Variante von Limit Hold'em Poker und wurde ja im obigen Abschnitt bereits detailliert beschrieben.

3. HOBBYBOT ist ein privat entwickelter Bot, der sich nur langsam auf seinen Gegner einstellt, aber spezielle Schwächen von Poki im 2-Spieler-Match ausnutzt.

4. JAGBOT ist ein sehr einfaches, formelbasiertes Programm, das sich auf rein rationales Spielen konzentriert ohne besondere Schwächen des Gegners herausfinden und ausnutzen zu können.

5. ALWAYS CALL BOT & ALWAYS RAISE BOT wurden hier als Extrembeispiel von schwachen Spielern genutzt, deren Spiel einfach vorhersagbar ist und von besseren Bots schnell durchschaut werden sollte. Sie wurden hier verwendet, um eine bessere Vergleichbarkeit der verschiedenen Bots in Bezug auf die Gewinnmaximierung gegen schwache Gegner zu gewährleisten.

Die in Tabelle2 zu beobachtenden Resultate des Experiments sind alle statistisch signifikant und werden in Form von durchschnittlich gewonnenen Big Blinds pro Hand dargestellt. Generell konnte man dem Experiment entnehmen, dass Vexbot nicht nur gegen alle anderen

Bots gewonnen hat, sondern von diesen jeweils auch noch am meisten Gewinne abschöpft. Auch bei Spielen gegen menschliche Spieler zeigte sich Vexbot größtenteils überlegen und gegen Experten mindestens ebenbürtig. Diese Ergebnisse haben allerdings keine so große Aussagekraft, da hier die menschlichen Spieler nicht die Geduld hatten eine statistisch signifikante Anzahl von Runden zu spielen und so glücksbedingte Schwankungen einen zu großen Einfluss haben. Beobachtet werden konnte allerdings ein Anstieg der Gewinnrate von Vexbot gegen menschliche Spieler nach ca. 200-400 Runden, was auf einen positiven Einfluss des Opponent Modelings schließen lässt.

Programm	Vexbot	Sparbot	Hobbot	Poki	Jagbot	Always Call	Alway Raise
Vexbot	-	+0.052	+0.349	+0.601	+0.477	+1.042	+2.983
Sparbot	-0.052	-	+0.033	+0.093	+0.059	+0.474	+1.354
Hobbot	-0.349	-0.033	-	+0.287	+0.099	+0.044	+0.463
Poki	-0.601	-0.093	-0.287	-	+0.149	+0.510	+2.139
Jagbot	-0.477	-0.059	-0.099	-0.149	-	+0.597	+1.599
Always Call	-1.042	-0.474	-0.044	-0.510	-0.597	-	0.000
Always Raise	-2.983	-1.354	-0.463	-2.139	-1.599	0.000	-

Table 4: Durchschnittlich gewonnene Big Blinds pro Hand

5. Fazit

Im Verlauf dieses Papers wurden einige Verfahren vorgestellt wie das Opponent Modeling realisiert werden kann. Dabei haben wir versucht aufzuzeigen, dass sowohl Poki wie auch Vexbot zwar sehr vielversprechende Ansätze, allerdings auch einige Unzulänglichkeiten mit sich bringen. Bei Vexbot zum Beispiel führt der aufwändige Aufbau des Suchbaumes dazu, dass diese Vorgehensweise für mehr als 2 Spieler kaum noch mit annehmbarer Rechenzeit umzusetzen ist. Das in Zusammenhang mit Vexbot vorgestellte Abstraktionsmodell ist allerdings eine sehr gute Möglichkeit trotz einer begrenzten Anzahl an Daten einen möglichst guten Eindruck des Gegnerverhaltens zu gewinnen. Poki andererseits bietet eine gute Herangehensweise, da hier durch spezielle Analysen die für die Einschätzung des Gegners signifikantesten Parameter ermittelt und auch nur diese in der Berechnung verwendet wurden. Doch obwohl die Entwicklung immer weiter voranschreitet, ist das Problem einer optimalen Poker-KI noch weit davon entfernt gelöst zu werden.Es bedarf noch vieler Verbesserungen, vor allem in Bezug auf das Opponent Modeling, da hier noch immer viele Informationen nur unzureichend genutzt werden. Poki zum Beispiel ignoriert vollkommen die in einem Showdown aufgedeckten Karten und die damit verbundenen Informationen über das Spielverhalten des Gegners. Hier könnte man zum Beispiel in dem Wissen seiner Handstärke seine in der selben Runde getroffenen Entscheidungen rückwirkend analysieren und so mögliche Fehler und Schwächen des Gegners aufdecken. Während andere Aspekte der Pokerprogramme möglicherweise nahezu Perfektion erreicht haben, bleibt die essentielle Funktion des Opponent Modelings mit großer Wahrscheinlichkeit auch in Zukunft eine große Herausforderung.

Abschließend kann man aber festhalten, dass die im Laufe dieses Papers vorgestellten Architekturen und Algorithmen bereits schon heute sehr starke Pokerprogramme hervorge-

bracht haben von denen menschliche Spieler mit Sicherheit vieles lernen können. Beobachtet man die aktuelle Entwicklung, liegt sogar die Vermutung nahe, dass Computerprogramme in absehbarer Zeit allen menschlichen Spielern in ihrer Spielstärke überlegen sein werden.

References

Billings, D., Papp, D., Schaeffer, J., & Szafron, D. (1998). Opponent modeling in poker. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pp. 493–499, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Davidson, A., Billings, D., Schaeffer, J., & Szafron, D. (2000). Improved opponent modeling in poker. In *Proceedings of The 2000 International Conference on Artificial Intelligence (ICAI'2000)*, pp. 1467–1473.

Davidson, A., Darse Billings, Jonathan Schaeffer, D. S. T. S. N. B. M. B., & Holte, R. (2004). Game tree search with adaptation in stochastic imperfect information games..