Entwicklung einer lernfähigen Bietfunktion für einen künstlichen Spadesspieler

Diplomarbeit
vorgelegt von
cand.-inform. Manfred Vonderheidt

30. November 2007

Technische Universität Darmstadt
Fachbereich Informatik
Fachgebiet Knowledge Engineering
Gebäude S2/02
Hochschulstraße 10
D-64289 Darmstadt

Betreuer und Prüfer:
Professor Johannes Fürnkranz

Ehrenwörtliche Erklärung

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

Darmstadt, 30. November 2007

Inhaltsverzeichnis

1 Einführung

Seit vielen Jahren wird intensiv an der Entwicklung von künstlichen Spielern für sehr unterschiedliche und bisher von Menschen dominierten Spielen geforscht. Die dabei erreichten Fortschritte unterscheiden sich jedoch von Spiel zu Spiel gewaltig. Während im Schach der damalige Weltmeister Kasparow bereits 1997 vom speziell für diesen Zweck entwickelten Schachprogramm *Deep Blue* besiegt wurde, und heutige Standardprogramme (auch auf Grund der mittlerweile verfügbaren Rechengeschwindigkeit) selbst den meisten erfahrenen Spielern überlegen sind, existiert z.B. für Go trotz ausdauernder Forschung bis zum heutigen Tag kein Programm, welches einem fortgeschrittenen Spieler ebenbürtig ist. Für Dame (in der Variante Checkers) wurde im April 2007 von Jonathan Schaeffer sogar bewiesen, dass bei einem perfekten Spiel die Partie immer unentschieden endet. [13]

Diese drei Brettspiele sind Beispiele für bereits gründlich erforschte Spiele, wenn auch mit unterschiedlichem Erfolg. Was alle drei auszeichnet ist ein hoher Bekanntheitsgrad und eine damit einhergehende weite Verbreitung. Doch auch bei Kartenspielen gibt es sehr interessante Themen, welche bisher unterschiedlich gut erforscht sind. Um sich näher mit einem Spiel oder einer bestimmten Unterart zu beschäftigen ist es notwendig, die Kartenspiele (zumindest grob) zu kategorisieren.

Eines der in Deutschland bekanntesten Kartenspiele ist Skat, weshalb ich Spiele mit der gleichen Charakteristik als skatähnlich bezeichne. Beispiele hierfür sind unter anderem Doppelkopf, Schafkopf, Herz, Spades und Bridge. Diese Spiele zeichnen sich durch folgende gemeinsame Grundregeln aus:

- Es wird in festen Runden gespielt.
- Die angespielte Farbe muss bekannt werden.
- Es existieren Stiche, welche aus den Karten einer Spielrunde gebildet werden.
- Optional existiert eine Trumpffarbe, welche hochwertiger als die anderen Farben ist.

Gute Gegenbeispiele zu dieser Spielart sind die zwei beliebten Spiele Poker und Rommé.

Einige bekannte Kartenspiele wurden bereits eingehend wissenschaftlich untersucht, u.a. Skat in [10] und [15], Herz in [11] und vor allem Bridge in [1], [3], [4], [7] und [16]. Neben diesen gibt es jedoch durchaus noch andere vielversprechende Kartenspiele, die bisher noch nicht näher betrachtet wurden.

Ein solches Spiel ist das bei uns eher unbekannte, in Nordamerika jedoch beliebte Spades. Grob betrachtet kann es als vereinfachtes Bridge betrachtet werden, da es in

beiden Spiele vorrangig um eine möglichst genaue Einschätzung der eigenen Karten zur Vorhersage der Stichanzahl geht. Trotz dieses gemeinsamen Charakters weichen die Spielstrategien beider Spiele deutlich voneinander ab, so dass die bisherigen Erkenntnisse nicht ohne weiteres auf dieses Spiel übertragen werden können.

Der interessanteste Spielteil bei Spades ist, wie auch bei Bridge, die Bietphase. In dieser Phase muss jeder Spieler eine grundlegende Entscheidung für den weiteren Verlauf der Spielrunde treffen. Von der Einschätzung der Güte des gegebenen Kartenblatts hängt maßgeblich der Spielerfolg ab; in der Tat kann eine vernünftige Kartenanalyse das Spielergebnis deutlich mehr beeinflussen als die darauf folgende Spielphase.

In der folgenden Abhandlung wird Spades im Allgemeinen und die Bietphase im Besonderen vorgestellt und analysiert. Dabei wird es mit anderen Kartenspielen verglichen und diesbezügliche Arbeiten vorgestellt, wobei im Vordergrund die mögliche Weiterverwendung bisheriger Erkenntnisse steht. Auf der Grundlage dieser Untersuchungen werden anschließend Algorithmen für eine möglichst optimale Spielweise des Spielers entwickelt. Darauf basierend wird ein künstlicher Spadesspieler implementiert und gegen Modifikationen der entwickleten Methoden, eine klassische statische Bietmethode und auch gegen unabhängige Spielgegner getestet.

Ziel dieser Arbeit ist eine fundierte Beurteilung darüber, mit welchem Aufwand und wie exakt die Stichanzahl bei Spades vorhergesagt werden kann, und welchen Einfluss eine Verbesserung der Bietmethode auf die Spielstärke insgesamt hat.

2 Vorstellung von Spades

2.1 Spielregeln

Wie bei praktisch allen Kartenspielen existieren für Spades unterschiedliche Regeln, insbesondere für die Berechnung der Punkte. Auch wenn der Kern immer gleich bleibt, kann das eigentliche Spiel und damit die Spielweise bereits durch eine Veränderung der Bepunktung deutlich variiert werden. Die im Nachfolgenden beschriebenen Regeln entsprechen zumeist dem unter [23] Beschriebenen, da auf Grund der Charakteristik der Artikelentstehung davon ausgegangen werden kann, dass es sich hierbei um die zur Zeit am weitesten verbreiteten Standardregeln handelt.

2.1.1 Allgemeine Beschreibung

Spades wird von vier Spielern mit 52 Karten gespielt. Diese sind in die vier Farben Kreuz, Pik, Herz und Karo unterteilt, wovon jede folgende 13 Karten enthält: Ass, König, Dame, Bube, 10, 9, 8, 7, 6, 5, 4, 3 und 2. Das eben beschriebene Kartenblatt entspricht einem einfachen Romméblatt ohne Joker.

Die vier Spieler sitzen im Idealfall jeweils in einem Winkel von 90° zu ihren Nachbarn. Die sich dabei gegenübersitzenden Spieler bilden jeweils ein Team.

Das Spiel basiert auf der Vorhersage der Spieler über die von Ihnen am Ende der Runde erreichten Stiche. Auf der Grundlage dieser Vorhersage und dem tatsächlichen Ergebnis werden nach einem weiter unten beschriebenen Schema Punkte vergeben. Das Ziel jedes Spielers ist das Erreichen der maximal möglichen Punkte für das Team durch ein mit dem Partner koordiniertes Spiel.

Zu Beginn des Spiels und jeder Runde werden die Karten gemischt und jeder Spieler erhält 13 Karten vom Kartengeber. Es ist keine bestimmte Art des Kartengebens vorgeschrieben; im Normalfall beginnt der Kartengeber mit seinem linken Nachbarn und teilt die Karten einzeln im Uhrzeigersinn aus. Nach jeder Runde wird das Amt des Kartengebers an den linken Nachbarn weitergegeben.

Nach dem Austeilen aller Karten werden von den Spielern jeweils ihre erwartete Stichanzahl angesagt. Anschließend werden alle Karten nach den folgenden Regeln offen ausgespielt.

Grundlegende Definitionen und Spielregeln

Stich Vier ausgespielte Karten bilden einen Stich, sofern hiervon jeder Spieler genau eine Karte regelkonform gespielt hat.

Runde Eine Spielrunde besteht aus 13 Stichen.

Spiel Ein Spiel umfasst mehrere Runden und endet mit dem Sieg einer Partei.

Wertigkeit der Karten Die Farben Kreuz, Herz und Karo sind gleichwertig. Pik hingegen ist hochwertiger als die anderen Farben und wird Trumpf genannt. Die Karten aller Farben sortiert nach absteigender Wertigkeit: Ass, König, Dame, Bube, 10, 9, 8, 7, 6, 5, 4, 3 und 2.

Luschen bezeichnen normalerweise Spielkarten, die keine Punkte zählen und deshalb für die Endabrechnung unwichtig sind. Bei Spades besitzen die einzelnen Spielkarten generell keinen Zählwert, da nur die Anzahl der erhaltenen Stiche entscheidend ist. Aus diesem Grund werden im Folgenden (entgegen der allgemeinen Definition) niedrige Karten (zwei bis zehn) als Lusche bezeichnet, mit denen im Normalfall kein Stich gemacht wird.

Aufspiel Das Aufspiel (Anspielen der ersten Karte eines jeden Stichs) für den ersten Stich hat der Spieler links des Gebers, danach immer der Gewinner des vorangegangenen Stichs. Es dürfen immer alle Karten der Farben Kreuz, Herz und Karo aufgespielt werden. Erst nachdem von einem Spieler in dieser Runde bereits eingestochen wurde oder der Aufspieler keine andere Farbe außer Pik mehr besitzt, ist auch ein Aufspielen von Pik gestattet.

Farbzwang Die angespielte Farbe (inkl. Pik) wird von den folgenden Spielern (Zweit-, Dritt- und Vierthand) gefordert. Sofern sie Karten dieser Farbe besitzen, muss davon eine gespielt werden (die Farbe muss *bekannt* werden). Ansonsten kann eine beliebige Karte gelegt werden. Das Ausspielen einer nicht geforderten Trumpfkarte wird *Einstechen* genannt.

Zuordnung eines Stichs Ein nur aus Farbkarten bestehender Stich wird dem Spieler zugesprochen, der die Karte mit der höchsten Wertigkeit der angespielten Farbe ausgespielt hatte. Wurde Trumpf angespielt oder eingestochen, erhält der Spieler der höchsten Trumpfkarte den Stich.

2.1.2 Bietphase

In dieser Phase sagen alle Spieler, beginnend beim Kartengeber, die Stiche voraus, die sie mit Ihren Karten erreichen wollen. Im Normalfall wird mindestens ein Stich angesagt. Neben der Bewertung der eigenen Karten können evtl. noch andere Informationen in die Ansage mit einfließen, wie z.B. die Anzahl der bisher angesagten Stiche, vor allem die des Partners. Allerdings kann die Aussagekraft dieser Daten nur bei bekannten Mitspielern begrenzt bestimmt werden, da hier bei menschlichen Spielern soziale Komponenten wie extreme Vorsicht oder Selbstüberschätzung eine große Rolle spielen. Die korrekte Bewertung der eigenen Karten stellt dabei das Hauptproblem dieser Arbeit dar.

Haben alle Spieler ihr Ansage getätigt ist die Bietphase beendet.

2.1.3 Spielphase

Nach den oben beschriebenen Regeln werden nun alle Karten ausgespielt. Der Spieler links des Kartengeber beginnt die Runde durch das Aufspielen einer Karte. Im Uhrzeigersinn folgen nun die restlichen Spieler. Die Spielphase endet, nachdem alle Karten aufgespielt und die daraus gebildeten 13 Stiche verteilt wurden.

2.1.4 Punktesystem

Die Punkte werden nach der abgeschlossenen Spielphase anhand der angesagten und wirklich gewonnenen Stiche berechnet. Dabei werden nur die Gesamtergebnisse des Teams betrachtet, d.h. die jeweils erhaltenen Stiche der einzelnen Spieler werden addiert und dann gemeinsam ausgewertet. Dabei spielt es ebenfalls keine Rolle, ob die einzelnen Spieler ihre Vorgabe für sich betrachtet erfüllt haben. Insgesamt können zwei Fälle unterschieden werden:

1. **Die angesagten Stiche wurden mindestens erreicht**
   Das Team erhält pro angesagtem Stich 10 Punkte gutgeschrieben. Jeder darüber hinaus erhaltene Stich wird mit einem zusätzlichen Punkt honoriert.

2. **Die angesagten Stiche wurden nicht erreicht**
   Dem Team werden pro angesagtem Stich 10 Punkte abgezogen.

Zusätzlich zu diesen normalen Punkten wird pro Team ein Konto mit Strafpunkten geführt. Hierauf werden alle Stiche addiert, die das jeweilige Team zu viel (gegenüber der Ansage) erhalten hat. Sobald das Konto zehn Punkte erreicht oder überschreitet, werden dem Team 100 Punkte abgezogen und die Strafpunkte um 10 reduziert.

Die Verrechnung der Strafpunkte wird vorgenommen, bevor das Spiel wegen der erreichten Punkte möglicherweise endet. Generell entsprechen die Strafpunkte immer der ersten Dezimalstelle des aktuellen Punktestands.

2.1.5 Spielende

Das Spiel endet, sobald ein Team 500 oder mehr Punkte erreicht hat, mit einem Sieg des entsprechenden Teams. Bei einem Punktestand von -200 oder weniger endet das Spiel ebenso, jedoch mit dem Sieg des gegnerischen Teams. Im Falle eines Punktegleichstands besteht die Möglichkeit, eine Entscheidungsrunde zu spielen oder das Spiel als unentschieden zu werten.

2.1.6 Sonderregeln

Nullspiel

Anstatt einer normalen Stichansage kann ein Spieler sich für ein Nullspiel entscheiden. In diesem Fall darf er, ähnlich dem Null beim Skat, keinen Stich erhalten, um zu gewinnen. Das Nullspiel hat einen Wert von 100 Punkten, die bei einem Sieg gutgeschrieben, bei einem oder mehreren gewonnenen Stichen jedoch abgezogen werden.

Bei der Wertung eines verlorenen Nullspiels gibt es mehrere Möglichkeiten, mit den gewonnenen Stichen des Nullspielers zu verfahren. Die Stiche des Nullspielers werden entweder

- ganz normal dem Partner hinzu gerechnet,
- komplett aus der Wertung genommen, können also weder zu wenige Stiche des Partners ausgleichen noch Strafpunkte geben oder
- zählen nur als Strafpunkte.

Doppelnull

Als Modifikation des normalen Nullspiels gibt es die Möglichkeit, vor dem Aufnehmen der Karten ein Doppelnull anzusagen. Gespielt wird nach den normalen Nullregeln, nur die Wertung ist mit 200 Punkten doppelt so hoch.

Da diese Option für das Erlernen der Wertigkeit von Kartenblättern jedoch denkbar ungeeignet ist, wird in der weiteren Arbeit nicht näher darauf eingegangen.

2.1.7 Strategien

Eine optimale Strategie lässt sich natürlich nicht durch die Angabe einiger Taktikempfehlungen erreichen. Jedoch kann durch diese eine vernünftige Spielweise nachvollzogen werden und die taktischen Grundkonzepte des Spiels werden offensichtlich.

Zu beachten ist hierbei allerdings, dass einige der genannten Punkte vor allem auf psychologische Schwächen der Gegner abzielen und deshalb nicht unbedingt auf ein Spiel mit oder gegen künstliche Spieler übertragbar sind.

- Sollte ein Gegenspieler seine Stichansage von den bereits angesagte Stichen abhängig machen, lässt sich dies ausnutzen um ihn zu verlocken, eine zu hohe Anzahl vorherzusagen. Dadurch kann das gegnerische Team möglicherweise leicht unter seine notwendigen Stiche gespielt werden.
- In einem koordinierten Spiel sollte genau darauf geachtet werden, welche Karten der Partner spielt. Wenn dieser hohe Karten abwirft, so möchte er wahrscheinlich keine weiteren Stiche mehr erhalten. Wirft er hingegen niedrige Karten ab, ist dies abhängig von den angesagten und bereits erhaltenen Stichen evtl. ein Zeichen dafür zu versuchen, den Gegner durch das Erhalten möglichst aller Stiche sein Spiel zu zerstören. Gleichfalls sollte erkannt werden, ob der Gegner einen wahrscheinlich eingeplanten Stich nicht, oder aber mit einer niedrigen Karte trotzdem erhält. Die eigene Spielweise kann für die Einhaltung der Ansage daraufhin angepasst werden.
- Die Strafpunkte erscheinen auf den ersten Blick oftmals relativ unerheblich, beherbergen jedoch ein großes Risiko und zugleich eine Angriffsmöglichkeit. Sind in der Bietphase noch viele Stiche offen, bietet es sich bei entsprechenden Karten an, sehr wenige Stiche anzusagen und das gegnerische Team mehr Stiche als geplant

erhalten zu lassen. Gerade durch die Art der Abrechnung kann damit durchaus ein sofortiger Sieg des gegnerischen Teams verhindert werden.

- Wenn der Partner ein Null spielt, sollte dessen Erfolg wegen der Wertigkeit die oberste Priorität erhalten, da der Gegner dieselbe Energie in die Vernichtung des Spiels setzen wird. Hohe Karten sollten zum Schutz des Partner aufgehoben werden, auch wenn dadurch ein Stich zu Gunsten der Gegner nicht gewonnen wird.

- Bei einem hohen punktemäßigen Vorsprung sollte auf Sicherheit gespielt werden, d.h. die führenden Spieler sollten die Stiche eher konservativ ansagen und kein riskantes Null spielen. Der Gegner muss dadurch zum Aufholen ein höheres Risiko eingehen und kann sich nicht darauf beschränken, das gegnerische Spiel zerstören zu wollen.

- Generell muss das Spiel immer auf die Mitspieler angepasst werden. Dieses Problem betrifft besonders auch die in dieser Arbeit vorgestellten Spielkomponenten, da die Erfolge bei bestimmten Gegnern, bedingt durch die jeweilige Spielstärke, nicht problemlos übertragbar sind.

3 Entwicklung und Implementierung eines Spadesspielers

In diesem Kapitel wird der theoretische Entwurf und die anschließende Umsetzung eines künstlichen Spielers für Spades vorgestellt. Zuerst werde ich den aktuellen Stand der Forschung bezüglich der Entwicklung künstlicher Kartenspieler aufzeigen und auch Vergleiche mit anderen Kartenspielen durchführen. Anschließend werden die Anforderungen einer Spades-KI an diese Arbeit festgestellt und auf dieser Basis die Entwicklung einer statischen KI beschrieben.

3.1 Aktueller Forschungsstand und Vergleiche mit anderen Kartenspielen

An dieser Stelle soll das soeben beschriebene Spades mit vier anderen, sehr bekannten Kartenspielen verglichen werden, um die konzeptionellen Unterschiede herauszuarbeiten. Diese betreffen vor allem die Entwicklung einer KI und die Möglichkeit, auf vorhandene Arbeiten zu diesen Spielen zurückzugreifen. Um im Rahmen dieser Arbeit zu bleiben, können die Spiele natürlich nur in ihren Grundzügen, d.h. ohne etwaige Sonderregeln bzw. -spiele vorgestellt werden; die Beschreibung erhebt deshalb keinen Anspruch auf Vollständigkeit.

3.1.1 Skat

Die offiziellen Skatregeln für Turniere finden sich auf den Seiten des Deutschen Skatverbands [9], einen guten ersten Überblick über die geläufigsten Varianten liefert Wikipedia [22].

Spielbeschreibung

Skat ist ein Kartenspiel für drei Personen und wird mit einem (eben nach diesem benannten) Skatblatt mit 32 Karten, bestehend aus vier Farben von je Ass bis 7, gespielt. Jeder Spieler erhält zehn Karten, die zwei übrigen werden nicht aufgedeckt und bilden den Skat. Keine Farbe ist vor dem Spiel als Trumpf festgelegt, zudem sind die Farben nicht gleichwertig. Am hochwertigsten ist Kreuz, gefolgt von Pik, Herz und schließlich Karo. Trotz dieser Reihenfolge kann ein Stich nur mit der höchsten Karte der angespielten Farbe oder einem Trumpf gewonnen werden.

Die vier Buben sind (meist) Trumpf, die Reihenfolge der restlichen Karten ist (anders als bei Spades): Ass, 10, König, Dame, 9, 8, 7. Jede dieser Karte hat einen bestimmten Wert:

- Das Ass ist elf Punkte wert,
- die 10 entspricht zehn Punkten,
- der König zählt vier Punkte,
- die Dame hat eine Wertigkeit von drei und
- der Bube wird mit zwei Punkten berechnet.
- Die drei Luschen 9, 8 und 7 sind (für die Abrechnung) wertlos.

Insgesamt ergeben sich damit 120 Punkte, wovon für den Sieg mind. 61 benötigt werden.

Bei Skat spielt ein Spieler alleine gegen die beiden anderen. Zu Beginn jeder Runde wird deshalb ermittelt, wer einzeln spielt. Dieser Vorgang wird *Reizen* genannt und hat gewisse Ähnlichkeit mit der Bietrunde von Spades, weshalb der Vergleich dieser beiden Spiele überhaupt interessant erscheint. Der Einzelspieler legt zu Beginn des Spiels eine Farbe als Trumpf fest oder spielt ohne Trumpffarbe (Grand). Ein Sonderspiel ist das Null, hier darf, wie bei Spades, kein einziger Stich gemacht werden (selbst wenn dieser null Punkte wert wäre).

Entscheidend für das Reizen sind die in den Handkarten enthaltenen Buben, wobei der Kreuzbube der hochwertigste, gefolgt von Pik-, Herz- und Karobube, ist. Sofern der Kreuzbube vorhanden ist, wird die Anzahl der lückenlos aufeinander folgenden Buben bestimmt. Fehlt der Kreuzbube jedoch, zählt stattdessen die Anzahl der fehlenden Buben bis zum evtl. vorhandenen hochwertigsten Buben. Sollte insgesamt kein Bube auf der Hand sein, werden noch die höchsten fehlenden Karten der gespielten Farbe mitgezählt. Die somit bestimmte Zahl wird anschließend um eins erhöht und bildet den ersten Faktor der sich anschließenden Multiplikation.

Im nächsten Schritt kommen die Wertigkeiten der Farben hinzu, wobei gilt, dass Kreuz 12, Pik 11, Herz 10, Karo 9 und ein Grandspiel 24 Punkte zählt. Für den konkreten Reizwert multipliziert jeder Spieler den Wert seiner bevorzugten Farbe mit dem zuvor ermittelten Faktor.

Beispiel: Es sind Kreuz-, Pik und Karobube auf der Hand; Herz würde sich zum Spielen anbieten. Vom höchsten Buben aus gesehen sind zwei auf der Hand, da der dritte (Herzbube) fehlt. Dies ergibt demnach 2+1=3 Punkte. Verrechnet mit Herz könnte also bis 30 gereizt werden.

Im Falle eines Siegs wird der maximale Reizwert für das eben durchgeführte Spiel als Punkte notiert; sollte das Spiel jedoch nicht gewonnen werden, wird die doppelte Menge von den Punkten des Spielers abgezogen. Bei einem Nullspiel gibt es keinen Trumpf, auch die Buben sind als normale Karten in ihrer Farbe eingereiht, ebenso wie die 10. Die Reihenfolge entspricht damit: Ass, König, Dame, Bube, 10, 9, 8 und 7. Da die Buben zu normalen Karten geworden sind, werden diese beim Reizen von Null nicht berücksichtigt. Ein normales Nullspiel hat einen Reizwert von 23.

Zu den eben erklärten Reizregeln gibt es noch spezielle Ergänzungen wie z.B. das Handspiel, Spielen mit offenen Karten oder das Ansagen von Schneider / Schneider schwarz

(die Gegner dürfen maximal 30 Punkte / keinen Stich bekommen). Hierauf werde ich an dieser Stelle nicht näher eingehen.

Nach der Reizphase nimmt der Einzelspieler den Skat auf und legt dann zwei Karten von seiner Hand verdeckt auf den Tisch. Dies können auch die selben Karten sein, die er im Skat aufgenommen hat. Durch diese Regelung ist es auch möglich, dass das geplante Spiel nicht mehr durchgeführt werden kann, da der Spieler sich durch das Finden z.B. eines Buben überreizt hat. Letztendlich zählen für das Spiel die Handkarten nach dem Aufnehmen des Skats. Diese Karten gehören dem Spieler, werden also zu seinen ansonsten gewonnenen Stichen addiert. Danach werden alle Karten regelkonform ausgespielt und schlussendlich die erzielten Punkte gezählt.

Unterschiede zu Spades

Der auffälligste Unterschied zu Spades dürfte das abweichende Spielziel sein. Während bei Spades die vorher selbst angesagten Stiche erreicht werden müssen, ist bei Skat das Erreichen von 61 Punkten notwendig. Durch die unterschiedliche Wertigkeit der einzelnen Karten variiert die Spielstrategie deutlich. So ist das *Abgeben* von einzelnen Stichen bei Skat kein Problem, diese können mit null Punkten sogar wertlos sein. Bei Spades spielt es hingegen keine Rolle, welche Karten in einem Stich liegen, es zählt nur das Erlangen desselbigen. Einzelne Karten zeichnen sich hier nur durch die Möglichkeit aus, damit einen Stich zu erlangen, sie besitzen jedoch darüber hinausgehend keine besondere Bedeutung.

Obwohl die Biet- und Reizphasen der zwei Spiele auf den ersten Blick sehr ähnlich zu sein scheinen, stehen dahinter verschiedene Konzepte. Bei Skat werden durch das Reizen zwei Dinge ermittelt:

1. Der Einzelspieler und
2. die von ihm gewählte Spielart (welche auch die Bepunktung maßgeblich mitbestimmt).

Jeder Spieler muss sich deshalb vor dem Spiel darüber im Klaren sein, bei welcher Spielweise er welche Siegeschance hat und wieviele Punkte er mit diesem Spiel erreichen (oder auch verlieren) kann. Generell sind hochwertigere Spiele nicht schwieriger zu gewinnen, jedoch hat ein Spieler selten die Auswahl zwischen verschiedenen Spielen mit der gleichen Gewinnwahrscheinlichkeit. Deshalb gibt es neben dem leichtesten oftmals noch ein weiteres mögliches Spiel, welches jedoch mit einem höheren Risiko, aber auch nicht selten deutlich höheren Punkten verbunden ist. Der Spieler muss nun das Optimum aus diesen beiden Eigenschaften ermitteln. Häufig kann das anvisierte Spiel nicht durchgesetzt werden, so dass der Spieler auf ein höherwertiges umsteigen oder das Reizen beenden muss (was von Anfang an eine mögliche Option darstellt). Der Spieler hat mit seinen Karten ein Fülle von Möglichkeiten und einen großen Entscheidungsspielraum darüber, was bzw. wie er dies genau erreichen möchte.

Die größte Abweichung zu Skat ist der Spielzwang und die damit verbundene geringere Gestaltungsmöglichkeit bei Spades; eine Wahl der Spielart findet praktisch nicht statt. Mit einem gegebenen Blatt muss auf jeden Fall gespielt werden, entscheidend ist hier

eine gute Abschätzung der Güte der Handkarten. Natürlich ist man auch bei Spades seinen Karten nicht bedingungslos ausgeliefert, jedoch sind die Variationsmöglichkeiten wesentlich geringer als bei Skat. Das Ziel ist es (abgesehen von der einzigen Ausnahme Null) immer, möglichst viele Stiche zu erreichen und diese vorher korrekt anzumelden. Es gibt keine Wahlmöglichkeit zwischen verschiedenen Punkten und Risiken, denn der Versuch, weniger als die maximal möglichen Stiche zu erreichen, hat keine Vorteile. Die Entscheidung des Spielers in der Bietphase ist fast immer auf zwei Gebote eingeschränkt (z.B. vier oder fünf Stiche ansagen), wobei hier objektiv betrachtet immer die sicherere Anzahl angesagt werden sollten, da der Mehrgewinn (zehn Punkte) in keinem Verhältnis zum möglichen Verlust (die komplette Ansage) steht. Bei Skat rechtfertigt sich ein höheres Risiko bei moderat höheren Punkten noch durch die Möglichkeit, überhaupt spielen zu können. Dies ist bei Spades jedoch nicht gegeben.

So gesehen hat der Spadesspieler nur die Möglichkeit, statt eines normalen Spiels ein Null anzusagen. Hier muss zum ersten Mal im Spiel zwischen den Punkten und der Siegeschance abgewägt werden. Da ein Nullspiel jedoch immer das Vielfache eines normalen Spiels wert ist, muss kein Vergleich zu letzterem durchgeführt werden. Sobald ein Nullsieg wahrscheinlich genug ist, sollte es auch gespielt werden, unabhängig von den ansonsten mit dem aktuellen Blatt möglichen Stichen. Von dieser Regel gibt es nur zwei Ausnahmen:

1. Der Partner hat in dieser Runde bereits ein Null angesagt. Zwei Nullspiele im Team werden fast nie gewonnen.

2. Zum Sieg fehlen nur noch sehr wenige Punkte (z.B. 20). Diese können mit einer normalen, niedrigen Anssage meistens wesentlich ungefährlicher erreicht werden als durch ein Null.

Bisherige Arbeiten und Fazit

Speziell zu Skat existieren bisher leider nur sehr wenige Arbeiten, konkret konnte ich zwei Stück ausfindig machen. Dies dürfte vor allem an der Verbreitung von Skat liegen, welche sich hauptsächlich auf den deutschsprachigen Raum begrenzt. Die erste Arbeit stammt von Sebastian Kupferschmid [10] und beschäftigt sich mit der „Entwicklung eines Double Dummy Skat Solvers“. Da Skat (wie auch Spades und die meisten Kartenspiele) ein Spiel mit unvollständigen Informationen ist (d.h. der Spieler kennt nicht alle Karten), wird in der Arbeit zur Lösung dieses Problems eine Monte-Carlo-Simulation verwendet. Auf diese Arbeit und auf die aus XSkat [5] bekannten Heuristiken baut Jan Schäfer [15] auf. Beide Autoren beschäftigen sich in ihrer Arbeit mit der eigentlichen Spielphase bei Skat, jedoch nicht intensiv mit dem Reizen. Für dieses Problem ist bisher nichts systematisches Vorhanden, hier besteht weiterhin Forschungsbedarf.

Die vorgestellte Monte-Carlo-Simulation zur Lösung von Skat wäre durchaus auch eine Option für eine Spades-KI. Allerdings ist die Anzahl möglicher Weltzustände bei Spades durch das verwendete Romméblatt um einiges größer als beim Skat, zudem soll im Vordergrund dieser Arbeit die Verbesserung der Bietphase stehen, und nicht die Entwicklung einer tadellos spielenden KI. Deshalb erscheint für die eigentliche Spielphase

der Einsatz einer auf Heuristiken basierten KI sinnvoller, welche ähnlich der XSkat- oder auch GGZ-KI [6] arbeitet.

Für die Bietphase bei Spades führen Arbeiten über künstliche Skatspieler leider nicht zu übernehmbaren Vorgehensweisen. Abgesehen davon unterscheiden sich beide Spielphasen deutlich voneinander. Beim Reizen müssen wesentlich mehr Faktoren berücksichtigt werden als dies bei Spades notwendig ist. Hier hat der Spieler im Endeffekt nur zwei Möglichkeiten: ein normales Spiel oder ein Null. Der gesamte Bietprozess ist wesentlich weniger dynamisch und kann für den Spieler leichter optimiert werden, da kaum eine Beeinflussung durch andere Spieler gegeben ist.

Insgesamt können trotz interessanter Ansätze leider wenige Ergebnisse von Skat für einen Spadesspieler genutzt werden. Dafür unterscheiden sich die Themen der vorhandenen Arbeiten sowie das untersuchte Spiel zu stark von den Zielen dieser Arbeit.

3.1.2 Herz

Eine genaue Beschreibung des Kartenspiels Herz mit Varianten für mehr als vier Spieler findet sich unter [19].

Spielbeschreibung

Herz ist ein Kartenspiel für (normalerweise) vier Spieler und wird wie Spades mit einem Romméblatt ohne Joker und gleicher Reihenfolge der Karten gespielt. Zu Beginn der Runde werden alle Karten ausgeteilt und von den Spielern aufgenommen. Direkt daran schließt sich eine Tauschrunde an, in welcher jeder Spieler drei seiner Karten seinem linken Nachbarn verdeckt zuschiebt. In der zweiten Runde wird nach rechts, in der dritten zum Gegenübersitzenden geschoben; jede vierte Runde entfällt das Kartentauschen. Das Spiel wird immer mit dem Aufspiel der Kreuz-Zwei begonnen, in diesem ersten Stich darf weder die Pikdame noch Herz gespielt werden (außer ein Spieler hat keine andere Karte auf der Hand).

Umgekehrt zu Skat ist das Ziel des Spiels, möglichst wenige Punkte zu sammeln. Jede Herzkarte zählt einen Punkt, zusätzlich ist die Pikdame 13 Punkte wert. In jedem Spiel werden folglich 26 Punkte verteilt. Das Spiel endet, sobald ein Spieler 100 Punkte oder mehr erreicht, der Spieler mit den wenigstens Punkten ist in diesem Fall der Sieger. Gelingt es einem Spieler, in einer Runde 26 Punkte zu erhalten, wird dies *Durchmarsch* genannt. In diesem Fall werden nur den drei anderen Spielern 26 Punkte aufgeschrieben.

Unterschiede zu Spades

Die Gemeinsamkeiten zwischen beiden Spielen sind recht klein, lediglich die grundsätzlich erlauben Spielzüge sind praktisch identisch. Bei Herz spielt jeder Spieler alleine für sich, es existieren keine Teams. Selbst bei einem versuchten (und vor allem auch erkannten) Durchmarsch ändert sich dies nicht. Zwar haben die restlichen Mitspieler durchaus ein großes Interesse daran, dass dieser misslingt, jedoch spielen sie auch in diesem Fall weiterhin als Konkurrenten gegeneinander.

Grundsätzlich ähnelt das Spiel einem Null, jedoch mit einer anderen Priorität, da das Erhalten von Stichen nicht unbedingt von Nachteil ist. Es können sogar darin enthaltene Punkte bewusst in Kauf genommen werden, um damit hohe Karten loszuwerden oder einen Durchmarsch zu vereiteln. Da jedoch vor allem gegen Ende einer Runde viele Punktekarten verteilt werden (die Spieler sind zu diesem Zeitpunkt häufig farbfrei), sollte auch hier versucht werden, möglichst keine Stiche mehr zu erhalten. Deshalb ist das Abwerfen einer normalen Farbkarte oftmals sinnvoller als das Verteilen von Punkten.

Vor der Runde steht lediglich die Entscheidung, ob ein Durchmarsch versucht werden soll. Sofern die Wahrscheinlichkeit auf einen Erfolg groß genug ist, sollte dies auch getan werden (ähnlich dem Nullspiel bei Spades). Ein großer Vorteil ist bei dieser Spielvariante, dass sie nicht vorher angemeldet werden muss, die Gegner also nicht von Anfang an versuchen werden, den Durchmarsch zu verhindern. Hierfür ist eine geschickte Spielweise notwendig, welche die Art des Spiels nicht frühzeitig verrät und trotzdem für den Erhalt jeder Punktekarte sorgt.

Bisherige Arbeiten und Fazit

Auf Grund der nur begrenzt zu treffenden Spielentscheidungen eignet sich eine Herz-KI sehr gut dafür, auf der Basis eines neuronalen Netzes implementiert zu werden, wie dies z.B. in [11] beschrieben wird. Die Erfolge einer solchen Methode sind zufriedenstellend, erreichen jedoch durch die eingeschränkte Sichtweite der Spielmethode nicht die Güte eines menschlichen Spielers. Dies war jedoch auch nicht Teil der Arbeit, es ging vielmehr um die generelle Möglichkeit das Spiel selbstständig zu erlernen.

Da das Augenmerk dieser Arbeit auf der Bietphase liegt, kann diesbezüglich nicht viel von Herz übernommen werden. Als Spiellogik scheint eine heuristische Methode auf Grund der größeren Wahlmöglichkeit bei Spades besser geeignet zu sein, jedoch bietet sich ein neuronales Netz durchaus für eine lernfähige Bietphase an.

3.1.3 Whist

Whist ist ein altes Kartenspiel und wird häufig als Vorgänger von Bridge bezeichnet. In der Tat existieren sehr viele und auch sehr unterschiedliche Variationen, deshalb würde ich Whist eher als Oberbegriff für eine ganze Spielfamilie auffassen. Die Beschreibung werde ich daher auch allgemein halten, angelehnt an die Variante *Oh Hell* [21], welche dem kommerziellen Kartenspiel Rage sehr ähnelt.

Spielbeschreibung

Whist kann mit unterschiedlich vielen Personen und verschiedenen Kartenblättern gespielt werden, üblich sind Skat- und einfache Romméblätter ohne Joker. Im Gegensatz zu den anderen hier beschriebenen Spielen werden nicht alle Karten zwangsläufig an die Spieler ausgeteilt, so dass häufig bis zum Rundenende nicht bekannt ist, welche Karten noch im Spiel sind. Üblich ist eine Veränderung der Kartenzahl pro Runden, beginnend bei einer Karte pro Spieler bis zu zehn Karten und wieder zurück (19 Spielrunden). Sieger ist der Spieler mit den meisten Punkten nach der letzten Runde.

Nach dem Austeilen der Karten müssen die Spieler nacheinander ansagen, wieviele Stiche sie mit ihren Karten erhalten werden. Üblich ist auch eine Regel, nach der die Gesamtsumme der Ansagen ungleich der Stichanzahl sein muss. Danach werden alle Karten ausgespielt, es herrscht auch hier Farbzwang. Am Ende der Runde erhält jeder Spieler seine gewonnenen Stiche als Punkte, bei einer Übereinstimmung mit seiner Ansage bekommt dieser zehn Zusatzpunkte. Im Gegensatz zu Spades darf die Ansage für die Sonderpunkte auch nicht überschritten werden, zudem gibt es immer nur zehn Punkte für eine Übereinstimmung.

Der Trumpf wechselt jede Runde und wird durch das Umdrehen der obersten Karte des Stapels nach dem Austeilen bestimmt, in Runde fünf und zehn entfällt der Trumpf.

Unterschiede zu Spades

Der größte Unterschied besteht in der gänzlich anderen Bepunktung des Spiels und der Tatsache, dass die Ansage exakt erfüllt werden muss. Durch diesen Grund, und auch das Unwissen über die sich im Spiel befindlichen Karten, wird die Ansage bzw. deren Erfüllung deutlich erschwert.

Bisherige Arbeiten und Fazit

Über die Whistversion *Oh Hell* existiert eine Arbeit von Jason Fong [2]. Dieser setzt zur Lösung des Spielproblems eine Monte-Carlo-Simulation ein, ähnlich den zu Skat bekannten Arbeiten. Die Bietphase wird heuristisch durch Kartenbewertungen und der Wahrscheinlichkeit auf einen Stich durch bestimmte Karten durchgeführt; für die Evaluierung wird von einem menschlichen Spieler geboten, die entwickelte KI beschränkt sich auf das eigentliche Spielen. Dies ist ein durchaus sinnvolles Vorgehen, da auch bei dieser Arbeit der Fokus auf der eigentlich Spielphase lag, nicht jedoch das Bieten optimiert werden sollte. Bedingt dadurch eignet sich dieser statische Ansatz der Bietmethode nicht zur Konstruktion einer lernfähigen für Spades.

3.1.4 Bridge

Besonders interessant erscheint der Vergleich von Bridge, genauer gesagt der am meisten gespielten Variante Kontrakt-Bridge, mit Spades. Auch wenn es nicht ganz korrekt ist, kann Spades doch als vereinfachtes Kontrakt-Bridge verstanden werden. Die genauen Regeln sind in [20] beschrieben.

Spielbeschreibung

Die Grundregeln von Bridge weisen eine große Ähnlichkeit mit Spades auf. Es ist wie dieses ein Spiel für vier Spieler, die je zwei Teams bilden. Gespielt wird ebenfalls mit einem Romméblatt und der gleichen Wertigkeit der Karten. Allerdings gibt es keine feste Trumpffarbe, diese wird erst durch die Bietrunde festgelegt. Zudem haben die Farben, ähnlich zu Skat, verschiedene Wertigkeiten (in absteigender Reihenfolge): Pik, Herz, Karo und Kreuz.

Vor dem eigentlichen Spiel gibt es eine Bietrunde, in welcher (ähnlich dem Skat) ein Einzelspieler und seine Spielart (Trumpffarbe oder Spiel ohne Trumpf) festgelegt wird. Obwohl die sich gegenübersitzenden Spieler ein Team bilden, spielt in Wirklichkeit der Spieler mit dem höchsten Gebot alleine. Der Partner dieses Spielers legt seine Karten offen auf den Tisch und muss nach den Anweisungen des Gegenübers ausspielen, deshalb wird er auch als Dummy bezeichnet.

In der Bietrunde macht der Kartengeber die erste Ansage, ihm folgen reihum die anderen Mitspieler. Ein Gebot besteht aus einer Stichanzahl und der gewünschten Trumpffarbe, wobei ein Spiel ohne Trumpf hochwertiger als ein Farbspiel ist. Der bietende Spieler muss ein höheres Gebot als sein Vorgänger abgeben oder passen; das erste Gebot muss dabei mindestens 7 Stiche umfassen. Sofern ein Gebot abgegeben wurde, endet die Bietrunde sobald dreimal hintereinander gepasst wurde, und der Spieler des zuletzt abgegebenen Gebots ist der Einzelspieler. Haben hingegen alle vier Spieler in der ersten Bietrunde gepasst, so wird das Spiel nicht durchgeführt.

Nachdem der Einzelspieler ermittelt wurde, werden die Karten ausgespielt, wobei der Spieler links des Alleinspielers beginnt.

Hat der Alleinspieler (gemeinsam mit seinem Partner) am Ende des Spiels seine Ansage erfüllt oder übertroffen, so ist die Partie gewonnen. Die Bepunktung ist jedoch deutlich komplexer als bei Spades, da nicht nur die erhaltenen Punkte, sondern darüber hinaus auch die Gefahrenlage des Spielers einberechnet wird, welche bereits zu Beginn des Spiels feststeht. Für eine genaue Beschreibung der Punkteregelung bei Kontrakt-Bridge verweise ich auf [18].

Unterschiede zu Spades

Die Abweichungen zu Spades, vor allem zu der Bietphase, sind trotz alledem recht groß. Während bei Spades jeder Spieler für seine eigenen Karten Stiche ansagt, und diese gemeinsam mit dem Partner erhalten muss, sagt ein Bridgespieler für seine Karten und die unbekannten seines Partners die Stiche an, welche er nur durch seine eigenen Spielentscheidungen (wenn auch für zwei Spieler) erhalten muss. Damit die Partner nicht komplett ohne Informationen auf die unbekannten Karten bieten, übernimmt die Bietphase auch offiziell die Funktion des Informationsaustauschs unter den Spielern. Gerade dies verkompliziert den Sachverhalt enorm, denn schließlich müssen sich die Partner vorher auf eine Art Protokoll verständigen. Dieses übernimmt nicht nur die Aufgabe, Informationen über die eigenen Karten zu übermitteln, sondern sogar den Partner aktiv über sein Blatt abzufragen. Laut Regelwerk müssen außerdem die Gegner über die Art der Kommunikation unterrichtet werden.

Da es im Gegensatz zu Spades viele professionelle Bridgespieler und -turniere gibt, ist auch die eigentliche Spielphase genau untersucht und in mehreren Büchern beschrieben worden. Hier dominieren mehrere bekannte Grundtaktiken mit etlichen Variationen. Dieses Wissen ist für ein erfolgreiches Bridgespiel unabdingbar, denn ein Spieler der Gegenpartei sollte die Spielweise seines Partner und auch Gegners einordnen können.

Bisherige Arbeiten und Fazit

Auf Grund der Bekanntheit des Bridgespiels wurde es schon oft untersucht und viele künstliche Spieler programmiert. Das erste wirklich anspruchsvoll spielende Bridgeprogramm war *GIB* von Matthew L. Ginsberg [7]. Auch auf sehr hohem Niveau spielt u.a. *Bridge Baron* [16].

Die in Bridge Baron eingesetzte Tignum-2-KI versucht die Spielweise eines menschlichen Spielers technisch nachzubilden. Hierfür wird kein Monte Carlo, sondern ein *Hierarchical Task-Network* (HTN) eingesetzt. Das Spiel wird mittels bekannter Bridgestrategien geplant; zur Erfüllung einer Strategie wird diese in einzelne Aufgaben und weitere Unteraufgaben zerlegt, welche zu erfüllen sind. Der wohl auffälligste Nachteil dieser Vorgehensweise ist das benötigte Expertenwissen, denn die Kenntnisse über Bridge müssen manuell einprogrammiert werden. Aus diesem Grund ist die erste Version von Tignum auch gescheitert, erst mit Tignum 2 gelang es den Autoren, durch den Einsatz verschiedener Hilfsmittel wie Makros und auf die Wiederverwendbarkeit des Codes hin konzipierter Komponenten, ihr Ziel zu erreichen.

Der Grundansatz dieses Projekts unterscheidet sich radikal von den gebräuchlichen Methoden, welche ein Spiel in einer gänzlich anderen Art erfassen und bewerten als ein menschlicher Spieler. Aus diesem Grund wird (sehr erfolgreich spielenden) Schachprogrammen oft vorgeworfen, ihr Spiel sei zwar effizient aber dennoch *hässlich*, da diese keine Strategie konsequent verfolgen, sondern jeden Zug einzeln bewerten. Dies mag aus der Sicht der Spieler durchaus nachvollziehbar sein, für viele Entwickler stellt jedoch gerade die Unabhängigkeit von Expertenwissen die Eleganz eines Projekts dar.

Bridge Baron 8 mit dem eben kurz vorgestellten Tignum-2-Code gewann im Jahr seiner Fertigstellung die Weltmeisterschaft im Computerbridge, die Spielstärke steht damit außer Frage.

Im Gegensatz dazu verwendet (der sogar zweimalige Weltmeister im Computerbridge) GIB die klassische Monte-Carlo-Simulation, um fehlende Informationen (dem aktuellen Spielverlauf entsprechend) zu ergänzen. Über diese somit vervollständigten Informationen baut es einen Suchbaum auf, in welchem schließlich mit einer optimierten Alpha-Beta-Suche eine optimale Spielstrategie gesucht wird.

Der Ansatz von GIB wurde in der Vergangenheit bereits mehrfach untersucht. I. Frank und D. Basin kritisieren in [3] die Art der Informationsgewinnung durch die Monte-Carlo-Simulation und die darauf ausgeführte Alpha-Beta-Suche, da diese von einem vollständigen Wissen aller Mitspieler und dadurch auch von ideal spielenden Gegnern ausgeht, was in der Realität nicht unbedingt zu der besten Spielstrategie führt. Als Reaktion haben die selben Autoren in [4] einen anderen Ansatz vorgestellt, der auf *Vector Minimaxing* und *Payoff-Reduction Minimaxing* basiert.

Durch die Suche nach einem optimalen Spielzug über (angeblich) vollständige Informationen entsteht ein weiterer, potentieller Fehler [12]: Der Spieler führt niemals Spielzüge zur Erweiterungen seiner Informationen über die Karten seiner Mitspieler durch, da er diese ja zu kennen glaubt. Dies jedoch ist eine weit verbreitete Strategie bei Bridge und auch oftmals unerlässlich.

GIB ist trotz aller Kritik an der Spielweise bis heute eines der stärksten Bridgepro-

gramme; viele der theoretisch möglichen Fehler scheinen sich in der Praxis auch auf Grund des seltenen Auftretens nicht sehr störend bemerkbar zu machen. Ginsberg selbst sieht die größte Schwäche seines Programms auch nicht beim eigentlichen Spiel, sondern in der Bietphase. Mit aus diesem Grund wurde in [1] sehr ausführlich und ausschließlich die Bietphase untersucht, da diese maßgeblich am Erfolg eines Spielers beteiligt ist. Die Autoren A. Amit und S. Markovitch stellen ein komplettes Framework zur Entscheidungsfindung vor, welches zwar speziell für Bridge entwickelt wurde, jedoch auch auf andere Spiele übertragbar sein sollte. Der beschriebene PIDM-Algorithmus (*Partial Information Decision Making*) basiert auf einer modellbasierten Monte-Carlo-Simulation zur Ergänzung fehlender Informationen. Die Besonderheit dieser Vorgehensweise liegt in der konsequenten Verwendung simulierter Mitspieler, sowohl des Partners als auch der Gegner, in der Bietphase, weshalb auch der Namenszusatz *modellbasiert* gewählt wurde. Dadurch soll eines der Hauptprobleme der Monte-Carlo-Simulation entfallen: die Annahme, dass alle Spieler über ein vollständiges Weltbild verfügen und perfekt agieren. Um die fehlenden Informationen über die Partnerkarten (und auch der Gegner) genauer ergänzen zu können, verwendet der Algorithmus die Informationen aus den Geboten der Mitspieler. Dies ist möglich, da mit den Ansagen Informationen über die eigenen Karten kommuniziert werden. Dabei werden zur Reduzierung der möglichen Kartenblätter sowohl die *anfragenden* als auch die *aussagenden* Gebote verwendet.

Notwendig wird dadurch eine Komponente, welche die Spielstrategie der Mitspieler erkennt und korrekt auswertet. Die Autoren haben deshalb ein lernfähiges Entscheidungsnetzwerk entwickelt, welches sich auf die Strategie des Partners einstellen soll. Anfänglich kann dieses manuell mit einer Strategie zur Reduzierung der möglichen Mitspielerkarten ausgerüstet werden, möglich ist aber auch ein Start ohne eine solche. Muster aus dem realen Training werden mit PIDM einer bestimmten Handlung zugeordnet und dem Entscheidungsnetz hinzugefügt. Dieses selbst wird schließlich auf Widersprüche untersucht und generalisiert, wofür ID3 benutzt wird. Durch diesen induktiven Lernvorgang an realen Trainingsdaten stellt sich das Netz auf die Mitspieler bzw. den Partner ein.

Die Knoten des Netzes, egal ob erlernt oder vorgegeben, nutzen Statusbedingungen, um die Zugehörigkeit eines Status zu einem Knoten zu bestimmen. Diese Bedingungen erfordern jedoch weiterhin Expertenwissen. Konkret werden in der Arbeit 35 bridgespezifische Merkmalsklassen der Handkarten verwendet.

Letztendlich wird vor jedem Gebot das Verhalten der Mitspieler abgeschätzt und der dadurch bedingte zukünftige Zustand ermittelt. Von allen Möglichkeiten wird schließlich die beste Option ausgewählt.

Übergreifend betrachtet ist Bridge mittlerweile sehr ausführlich und auch erfolgreich in diversen Arbeiten behandelt worden. Die eigentliche Spielphase ist sehr ausgereift, der Schwachpunkt aller Programme war die Bietfunktion. Die Arbeit von Amit und Markovitch könnte dies geändert haben, denn ihr Ansatz zeigt (nach ausreichendem Training) deutlich bessere Ergebnisse als z.B. GIB. Dies ist jedoch mit einem verhältnismäßig hohem Aufwand verbunden, wie er sonst nur für die eigentlichen Spielroutinen notwendig ist.

3.1.5 Fazit

Nach der Untersuchung verschiedener Arbeiten zu unterschiedlichen Kartenspielen hat sich herausgestellt, dass meist nur die Spielphase eingehend untersucht wurde. Diese stellt auch meistens die größere Herausforderung dar, doch letztendlich entscheidet über den Spielerfolg das schwächste Glied in der Kette, und dies ist oftmals das Bieten bzw. Reizen.

Der aktuelle Stand der Forschung bzw. die fast ausschließlich anzutreffende Technik bei Spielen mit unvollständigen Informationen ist die Monte-Carlo-Simulation zur Vervollständigung der fehlenden Daten, auf denen anschließend ein Suchbaum aufgespannt wird. Viele Arbeiten beschäftigen sich mit der Beschneidung der Suchbäume, um diese in vertretbarer Zeit zu durchsuchen bzw. heuristisch auszuwerten. Ein Grund für dieses Vorgehen ist sicherlich das nicht oder nur begrenzt benötigte Expertenwissen, welches für Ansätze wie HTN obligatorisch ist.

Für die eigentlichen Bietfunktionen ist in der Literatur leider keine Systematik zu entdecken. Die meisten Modelle behandeln diese eher stiefmütterlich mit den Methoden ihrer Spielphasen. Lediglich die letzte bei Bridge vorgestellte Arbeit widmet sich ganz diesem Thema.

Spades stellt deutlich geringere, aber vor allem auch andere Ansprüche an eine Bietfunktion als z.B. Bridge. Eine direkte Übernahme der in [1] vorgestellten Methoden ist deshalb wenig sinnvoll.

3.2 Anforderungen an einen künstlichen Spadesspieler

An eine für diese Arbeit ideale KI müssen keine besonders hohen Ansprüche bezüglich der Spielstärke gestellt werden. Wesentlich wichtiger ist eine nachvollziehbare Spielweise mit kontinuierlich gleich hohem Erfolgswert gegen den gleichen Gegner. Zudem sollte sie unabhängig von den angesagten und bereits erhaltenen Stichen immer versuchen, so viele Stiche wie möglich zu erhalten. Dies ist zwar für den Erfolg in einem echten Spiel von Nachteil, führt für die Konditionierung der Lernkomponente jedoch zu dem Vorteil, dass die Anzahl der wirklich erhaltbaren Stiche registriert wird, welche nicht bereits durch strategische Entscheidungen beeinflusst wurde. Diese Daten sind als Ausgangsmaterial für zukünftige Entscheidungen besser verwertbar.

3.3 Frei verfügbare Spadesspieler

Als Basis für diese Arbeit kamen drei freie Projekte in Frage:

1. **SARC 1.1** [8]
   Das Projekt *Spades Artificially Reasoning Computer* wurde 2001 begonnen und bereits kurze Zeit darauf nicht mehr weiterentwickelt. Dementsprechend befindet es sich in einer sehr frühen Phase mit wenigen Funktionen, zudem existiert keine modulare Trennung zwischen KI und grafischer Oberfläche. Als Grundlage für diese Arbeit eignete sich dieses Projekt dadurch nicht.

2. **KardsGT 0.4.0** [14]
   KardsGT ist ein relativ junges Projekt und beinhaltet eine Sammlung von Kartenspielen. Es verfolgt interessante Ansätze wie z.B. Computergegner mit verschiedenen Spielcharakteristiken. Leider zeigte die Spades-KI jedoch eklatante Spielschwächen, wodurch sie für diese Arbeit uninteressant wurde.

3. **GGZ 0.0.14** [6]
   Das Projekt *GGZ Gaming Zone* stellt eine Infrastruktur für diverse Spiele bereit, um diese über das Internet gegen menschliche und künstliche Gegner zu spielen. Das Konzept ist sehr modular, es existieren neben dem Server Clients für verschiedene Spiele in mehreren Programmiersprachen. Die Spielstärke der Spades-KI des Servers ist passabel.

Obwohl ich die GGZ-Infrastruktur, speziell den Server und die Kommunikationsprotokolle, zur Evaluierung benutze, habe ich mich gegen eine Verwendung der vorhandenen KI entschieden. Eine Rolle spielte die Architektur mit der im Server integrierten KI, wichtiger war jedoch die Spielweise derselbigen. Diese ist an manchen Stellen bereits zu ausgefeilt. So kann die KI mit mehreren Aggressionsleveln spielen. Beispielsweise versucht sie dadurch weniger Stiche als möglich zu bekommen, wenn ihre Vorhersage der Stichanzahl bereits erfüllt ist. Ebenso ist es vorgesehen, dem Gegner viele Strafpunkte zu bescheren. Eine KI zur Klassifizierung bestimmter Kartenblätter sollte jedoch immer gleich und maximal aggressiv spielen, d.h. in jeder Situation (abgesehen vom Nullspiel) auf die maximale Stichanzahl, damit Verfälschungen der Lernergebnisse ausgeschlossen werden. Die GGZ-KI ist damit für die eigentliche Lernphase nicht perfekt geeignet, stellt jedoch einen interessanten KI-Gegner für die Evaluation der Arbeit dar.

Da keine frei zur Verfügung stehende Spades-KI alle Anforderungen erfüllt, erschien es am sinnvollsten, eine eigene zu implementieren.

3.4 Spielphase

In den folgenden Unterpunkten werden die einzelnen Komponenten der KI detailliert beschrieben. Hierfür werden jeweils die Anforderungen aufgezeigt und anschließend die konkrete Implementierung der verschiedenen Spielmodi vorgestellt.

Die KI basiert auf statischen Entscheidungsbäumen mit heuristischen Annahmen für eine möglichst optimale Spieltaktik in den jeweiligen Situationen. Zuerst wird jeweils der entsprechende Graph dargestellt und anschließend die einzelnen Entscheidungen sowohl hinsichtlich einer sinnvollen Spielweise als auch die technische Umsetzung näher beschrieben.

Im folgenden Abschnitt werde ich die Spielweise der KI für die einzelnen Spielarten näher erläutern. Dabei werde ich nicht starr die Entscheidungsbäume beschreiben, sondern die dahinterliegende Spielidee skizzieren. Die genauen Entscheidungen sind durch die Graphen ersichtlich.

Wird eine Spielentscheidung bei identischen Ausgangswerten getroffen (z.B. gleiche Karten in zwei Farben), erfolgt dies immer mit Hilfe einer Zufallskomponenten.

3.4.1 Definitionen häufig verwendeter Begriffe

Wird im Folgenden von einer Farbe gesprochen, ist damit im Allgemeinen nur Kreuz, Herz und Karo gemeint; Pik wird wegen der Übersichtlichkeit als Trumpf bezeichnet.

Stehende Karte Eine Karte wird als stehend bezeichnet, wenn sie die höchste Karte der Farbe im Spiel ist. So kann durchaus eine Dame stehen, wenn Ass und König bereits gefallen sind. Sind zu einer stehenden Karte noch eine oder mehrere Karten mit direkt folgender Wertigkeit vorhanden, können diese auch als stehend bezeichnet werden, z.B. Ass, König und Dame auf der Hand. In diesem Fall kann auch die Niedrigste davon gespielt werden.

Sobald bekannt ist, dass ein Gegenspieler in dieser Farbe frei ist, kann keine Karte dieser Farbe mehr als stehend eingestuft werden. Der Partner darf für diese Klassifikation frei sein, da er im Normalfall nicht einstechen muss. Im Moment nicht stehende Karten können durchaus im Verlauf der Runde noch diesen Status erlangen, nachdem die höheren Karten bereits gefallen sind.

Voraussichtlich nicht stehende Farbe Analog zur obigen Definition der stehenden Karten kann eine Farbe als voraussichtlich nicht stehend bezeichnet werden, sofern keine Karte dieser Farbe im Verlauf dieser Spielrunde als stehend eingestuft werden kann bzw. muss. Mit dieser Klassifikation soll die Entscheidung erleichtert werden, welche Farbe am ehesten abgeworfen werden kann. Sie schließt nicht aus, dass mit einer Karte dieser Farbe ein Stich gemacht oder sie zukünftig doch als stehend bezeichnet werden kann.

Da bei einer gleichmäßigen (idealen) Kartenverteilung jeder Spieler drei (bzw. einer vier) Karten einer Farbe erhält, ist ein normaler Farbenstich nur dreimal möglich (siehe 4.1.1). Auf Grund dieser Annahme gilt eine Farbe als voraussichtlich nicht stehend, sofern eine der drei folgenden Bedingungen erfüllt ist:

1. Es ist weder Ass noch König noch Dame vorhanden. (Aus Konsistenzgründen allerdings mindestens eine niedrigere Karte).
2. Der König ist die einzige Karte dieser Farbe.
3. Es befindet sich nur die Dame sowie maximal eine weitere Karte auf der Hand.

Abwerfen In vielen Situationen, in denen nicht bekannt werden muss, ist es nicht möglich oder nicht sinnvoll, einen (höheren) Trumpf auszuspielen. Demzufolge muss entschieden werden, welche Karte am besten abgegeben wird. Dabei gilt es zu beachten, keine Karten auszuspielen, durch die evtl. in der Zukunft stehende Karten dieser Farbe gefährdet werden. Einfaches Beispiel: In einer Farbe ist nur König und Bube vorhanden, das Ass jedoch noch nicht gespielt. Durch das Abwerfen des Buben muss beim Anspiel des Ass' mit dem König bekannt werden und dieser kann dadurch keinen Stich mehr gewinnen.

Vorteilhaft ist jedoch die Möglichkeit, durch das gezielte Abwerfen in einer Farbe frei zu werden, um daraufhin einstechen zu können. Im Endeffekt wird nach diesen Prioritäten abgeworfen:

1. Von den voraussichtlich nicht stehenden Farben mit den wenigsten Karten auf der Hand wird die Niedrigste ausgespielt.
2. Es wird die niedrigste Karte der Farbe mit den wenigsten Handkarten gespielt.
3. Der niedrigste Trumpf wird gespielt.

Abwerfen bei Null Das Abwerfen bei Nullspielen unterscheidet sich stark vom eben beschriebenen normalen Abwerfen, da der Spieler ein gegenteiliges Ziel damit verfolgt. Prinzipiell soll immer die gefährlichste Karte abgeworfen werden, d.h. die Karte, mit der am wahrscheinlichsten ein unerwünschter Stich gemacht wird. Letztendlich muss natürlich nur entschieden werden, von welcher Farbe (nicht Trumpf) die höchste Karte abgeworfen wird. Jedoch ist die Bewertung des Risikos einer Farbe nicht trivial: Eine blanke fünf kann durchaus gefährlicher sein als ein Ass, in dessen Farbe noch drei niedrige Karten vorhanden sind.

Im Endeffekt wird für diese Funktion die gleiche Einschätzung genutzt, wie sie auch bei der statischen Nullbietmethode Verwendung findet: Von allen Farben wird der Durchschnittswert der vier niedrigsten Handkarten berechnet und schließlich die größte Karte der Farbe mit dem höchsten Wert gespielt (siehe auch 3.5.2).

Eine weitere Idee war die Einschränkung auf Farben, in welchen die Gegner noch Karten besitzen, da nur dann eine Gefährdung für den Nullspieler bestehen kann. In der Praxis hat sich diese Unterscheidung jedoch nicht durch höhere Spielleistungen bestätigt, weshalb sie wieder entfallen ist.

Eine besondere Rolle spielt das Abwerfen von Trumpfkarten bei Nullspielen, da diese besonders leicht unerwünschte Stiche verursachen. Gefahrlos kann eine Trumpfkarte abgeworfen werden, wenn der Spieler nicht bekennen muss und bereits mit einem höheren Trumpf eingestochen wurde. Trotzdem kann es auch in dieser Situation sinnvoller sein, eine gefährliche Farbkarte abzuwerfen, da z.B. nur noch ein kleiner Trumpf mit geringem Risiko auf der Hand ist.

Letztendlich spielt es jedoch auch an dieser Stelle keine große Rolle, nach welchem Kriterium eine Trumpfkarte abgeworfen wird. Nach einigen Versuchen mit mehr oder weniger komplizierten Vergleichen der Hand- mit den Freikarten in Pik hat sich herausgestellt, dass das Abwerfen von Trumpf bei jeder Gelegenheit die Spielleistung insgesamt nicht nachweislich beeinflusst.

Freikarten Als Freikarten werden alle Karten einer Farbe bezeichnet, die sich noch im Spiel und nicht auf der Hand des Spielers befinden. Durch deren Anzahl kann abgeschätzt werden, wie hoch die Wahrscheinlichkeit ist, dass einer der Mitspieler keine Karte dieser Farbe mehr besitzt.

3.4.2 Bestimmung und Auswahl der Spielart

An dieser Stelle wird durch den Spieler ermittelt, ob er selbst oder einer der Mitspieler ein Null spielt. Ist dies der Fall, wird für den aktuellen Spielzug die der Situation entsprechende Funktion aufgerufen.

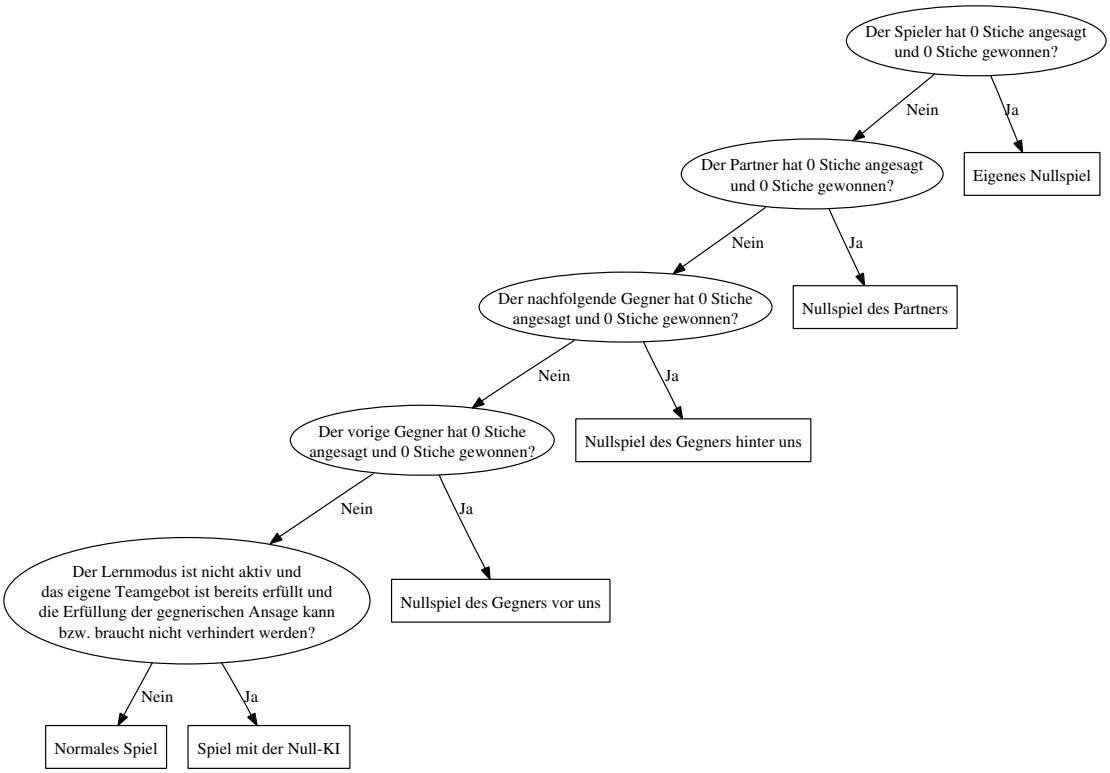

Abbildung 3.1: Bestimmung der aktuellen Spielart

Die Überprüfung der Spielart ist nicht weiter kompliziert. Hat der Spieler selbst null Stiche angesagt und auch bisher keinen Stich erhalten, so spielt er als aktiver Nullspieler. Ist dies nicht der Fall, wird zuerst der Partner und dann die zwei Gegner auf die gleiche Art überprüft. Der letzte Knoten ist eine Optimierung der Spielleistung, dessen Entscheidung nur dann positiv ausfallen kann, wenn die KI nicht im Lernmodus verwendet wird. In diesem Fall wird überprüft, ob das Team des Spielers bereits sein Gebot erfüllt hat und es zudem nicht möglich ist, die gegnerische Partei an der Erfüllung der Ansage zu hindern bzw. dies nicht mehr notwendig ist. In diesem Fall versucht der Spieler, keine weiteren Stiche mehr zu erhalten, wozu er mit der KI des aktiven Nullspielers weiterspielt. Sind letztendlich alle Vergleiche negativ, wird ein normaler Spielzug vollzogen.

Sollte ein beliebiger Spieler ein Null angesagt haben, jedoch durch das Erhalten eines Stichs dieses bereits verloren haben, wird die laufende Spielrunde als Standardspiel beendet. Durch die Reihenfolge der Überprüfungen wird auch sichergestellt, dass im Falle zweier Nullspiele (vom Partner und eines Gegners) der Partner unterstützt wird. Nicht unterschieden wird hingegen, wenn beide Gegner ein Null spielen. Dies sollte in der Praxis jedoch nicht vorkommen bzw. ein Doppelsieg auch ohne explizite Prüfung verhindert werden.

Zu Beginn jedes Entscheidungsprozesses muss zuerst die Position des Spielers bezüglich des Aufspielers bestimmt werden. Oftmals werden zwei Positionen (z.B. Dritt- und Vierthand) identisch behandelt, was jedoch von der Spielart abhängt.

3.4.3 Standardspiel ohne Null

Dieser Abschnitt beschreibt die Spielweise der KI bei einem Spiel, in dem kein Spieler ein Null angesagt hat.

Ist der Spieler am Aufspiel, so spielt er, wenn möglich, eine stehende Karte aus. Ist keine solche vorhanden, ist das Spielen einer Farbe, in welcher der Partner frei ist, auch eine gute Option. Ist beides nicht möglich, so wird eine unwichtige Karte (nach den Abwurfregeln) gespielt.

Die Entscheidungen für Zweit- und Dritthand sind identisch und wurden deshalb zusammengefasst. Wenn der Spieler bekennen muss, so lässt er den Stich laufen, wenn dieser bereits dem Partner mit einer stehenden Karte gehört. Ansonsten wird er (sofern nicht gestochen wurde) den Stich mit einer stehenden Karte übernehmen. Ist auch dies nicht möglich und es gibt eine höhere Karte dieser Farbe auf der Hand, so wird die zweithöchste gespielt, bei nicht stehenden Farben die höchste Karte. Anderenfalls wird einfach die niedrigste Karte ausgespielt.

Muss in dieser Situation jedoch nicht bekannt werden, so wird eine Karte genau dann abgeworfen, wenn der Stich dem Partner mit einer stehenden Karte gehört und maximal elf Karten dieser Farbe bereits gefallen sind. Wurde bisher nicht eingestochen und es ist Trumpf auf der Hand, wird davon ein mittlerer gespielt. Ansonsten wird der gespielte Trumpf, sofern möglich, überboten. In allen anderen Fällen wird eine Karte abgeworfen.

Bei der Position an letzter Stelle ist die Entscheidung natürlich relativ leicht, da die eigene Karte den Abschluss des Stichs bildet. Vereinfacht gesagt wird der Stich an dieser Stelle immer mitgenommen, sofern er nicht dem Partner gehört (oder er nicht übernom-

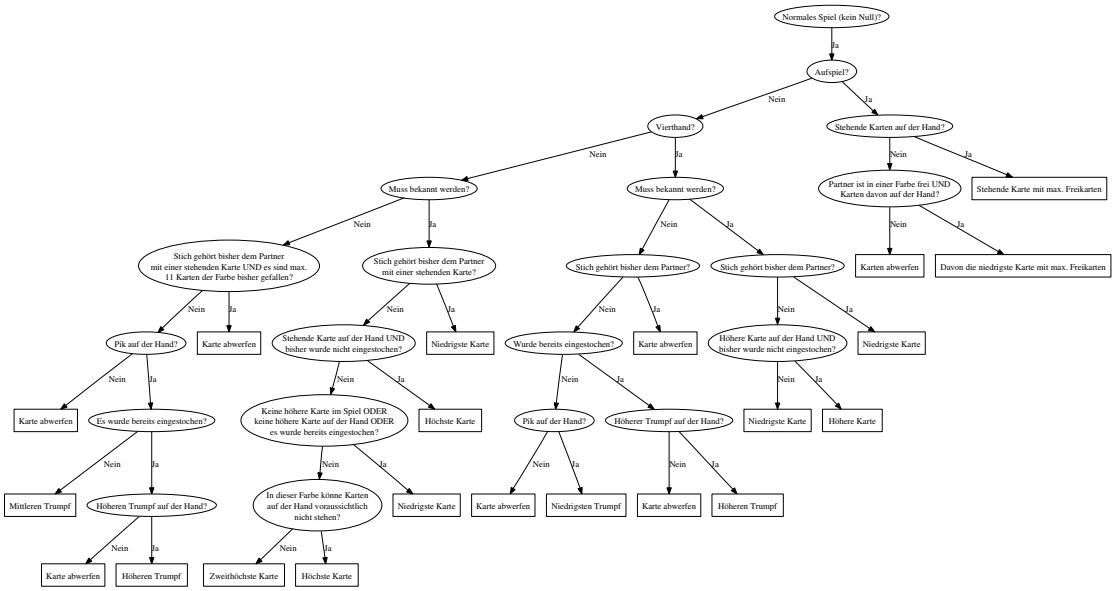

Abbildung 3.2: Statischer Entscheidungsbaum für ein Spiel ohne Null

men werden kann). Ansonsten wird die niedrigste Karte der geforderten Farbe gespielt bzw. eine Karte abgeworfen.

3.4.4 Eigenes Nullspiel

Nach folgendem Algorithmus wird gespielt, sofern der Spieler ein Null angesagt und auch noch keine Stiche erhalten hat.

Das Aufspiel bei Null sollte nur einmal beim ersten Stich vorkommen, ansonsten hätte der Spieler bereits verloren. Da es nicht klug ist, gleich die niedrigste Karte zu spielen, wird in diesem Fall von der Farbe mit der kleinsten Varianz die mittlere Karte gespielt.

Ansonsten wird im gesamten Spielverlauf immer versucht, unter der momentan führenden Karte im Stich zu spielen. Ist dies nicht möglich, so wird die niedrigste Karte der geforderten Farbe gespielt. Muss bekannt werden und es wurde bereits eingestochen, so kann die höchste Karte dieser Farbe gespielt werden. Wenn nicht bekannt werden muss und bereits eingestochen wurde, so wird (sofern dies sinnvoll ist) ein Trumpf abgeworfen.

3.4.5 Spielweise als Partner des Nullspielers

Als Partner des Nullspielers ist es das Ziel, den Erfolg des Nullspiels zu ermöglichen. Die selbst angesagten Stiche werden wegen der hohen Wertigkeit des Nullspiels diesem Ziel untergeordnet.

Beim Aufspiel wird (sofern dies erlaubt ist) ein stehender Trumpf aufgespielt. Dadurch bleibt der Spieler am Anspiel, was für den nullspielenden Partner immer von Vorteil ist. Als zweite Möglichkeit wird die stehende Karte einer Farbe gespielt und drittens eine

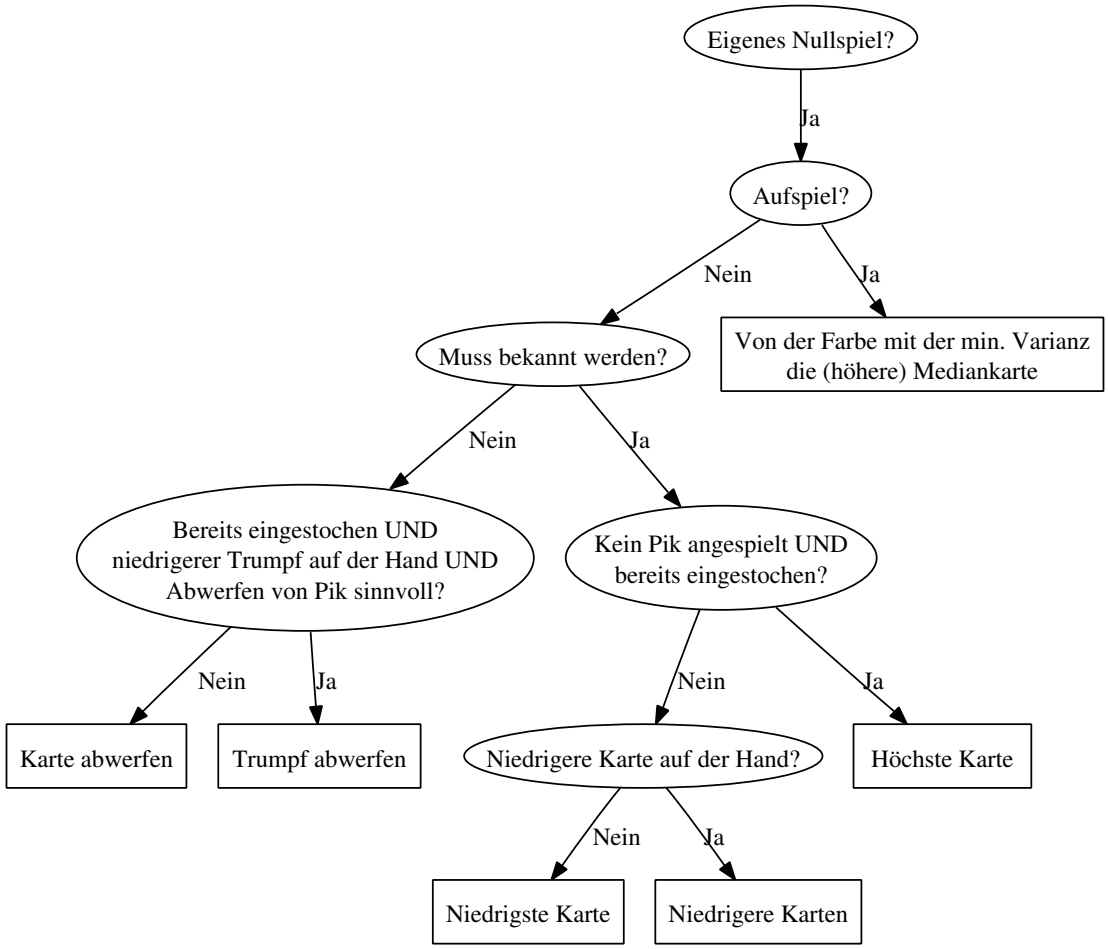

Abbildung 3.3: Statischer Entscheidungsbaum für den Nullspieler

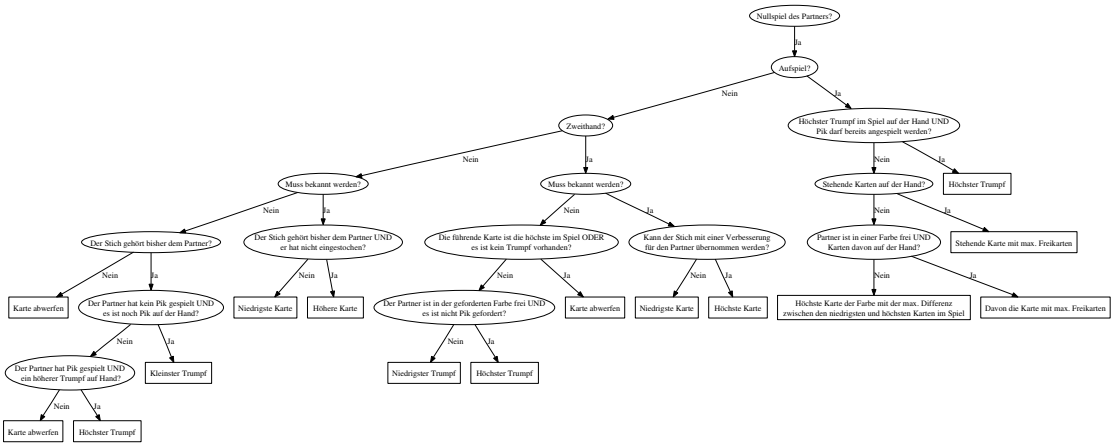

Abbildung 3.4: Statischer Entscheidungsbaum für den Partner des Nullspielers

Farbe, in welcher der Partner frei ist. Ist das alles nicht durchführbar, so wird die höchste Karte der Farbe mit der größten Differenz zwischen der kleinsten und höchsten Karte gespielt.

Bei diesem Entscheidungsbaum kann die Dritt- und Vierthand zusammengefasst werden, da es keine Rolle spielt, ob noch ein Gegner folgt. Allgemein gesagt wird jeder Sich, in dem der Nullspieler führt, mit der kleinstmöglichen Karte übernommen, da die höheren Karten noch gebraucht werden könnten. Zum Abwerfen wird die gleiche Methode wie beim aktiven Nullspiel verwendet.

3.4.6 Spielweise als Gegner des Nullspielers

Obwohl sich die Spielweise der zwei Gegnertypen (unterschieden wird zwischen dem Gegner vor und hinter dem Nullspieler) stark ähnelt, ist sie dennoch nicht identisch. Aus diesem Grund gibt es für diese zwei Fälle getrennte Entscheidungsgrundlagen.

Generell versuchen die Gegner immer, einen Sieg des Nullspielers zu verhindern und nehmen für dieses Ziel auch eine Unterschreitung ihrer Ansage in Kauf.

Der Gegner hinter dem Nullspieler

Für das Aufspiel wird eine Farbe gesucht, in welcher der Gegner frei ist, der Nullspieler jedoch (wahrscheinlich) nicht. Alternativ wird nach der Farbe gesucht, in der der Nullspieler am wahrscheinlichsten noch Karten hat. Von der ausgewählten Farbe wird in beiden Fällen die Mediankarte gespielt.

Hat der Nullspieler aufgespielt, so wird versucht, unter der angespielten Karten zu bleiben oder die jeweils höchste der geforderten Farbe zu spielen. Muss nicht bekannt werden, so wird eine Karte nach den gleichen Regeln wie beim aktiven Nullspiel abgeworfen.

Dritt- und Vierthand sind für diesen Spieler zusammengefasst. An dieser Stelle war der Nullspieler bereits am Zug. Muss die angespielte Farbe bekannt werden, stellt sich die

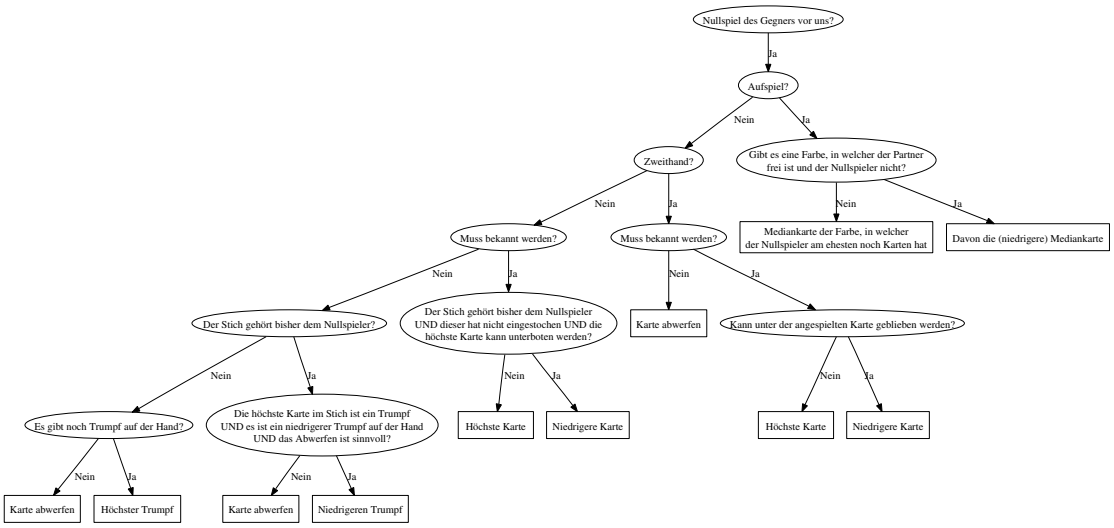

Abbildung 3.5: Statischer Entscheidungsbaum für den Gegner hinter dem Nullspieler

Frage, ob die höchste Karte im Stich vom Nullspieler stammt und dieser nicht eingestochen hat. In diesem Fall wird, sofern vorhanden, eine niedrigere Karte gespielt. Ansonsten legt der Spieler die höchste Karte dieser Farbe.

Wenn nicht bekannt werden muss, kann entweder abgeworfen oder eingestochen werden. Sofern der Stich nicht dem Nullspieler gehört, wird mit dem höchsten Trumpf gestochen. Ist der Nullspieler im aktuellen Stich mit einem Trumpf führend, wird möglichst ein kleinerer gespielt. In den beiden anderen Fällen wird eine Karte abgeworfen.

Der Gegner vor dem Nullspieler

Das Anspiel des Gegners vor dem Nullspieler ist dem bereits Beschriebenen sehr ähnlich, nur dass statt der Mediankarte die niedrigste Karte der gewählten Farbe gespielt wird.

An der zweiten und auch dritten Stelle wird zuerst unterschieden, ob bekannt werden muss. Ist dies nicht der Fall, so wird eine Karte abgeworfen. Ansonsten wird unterschieden, ob bereits eingestochen wurde (und kein Pik angespielt wurde) oder nicht. Wurde eingestochen, so kann die höchste Karte der geforderten Farbe gespielt werden. Andernfalls wird die Karte mit der geringsten Differenz zur höchsten Karte im Stich gespielt, wobei bei gleichem Abstand die niedrigere Karte bevorzugt wird.

Als Vierthandspieler wird genauso wie beim Spiel hinter dem Nullspieler vorgegangen: Wenn bekannt werden muss und der Stich dem Nullspieler gehört, so wird möglichst unter dieser Karte geblieben oder die höchste Karte gespielt. Ansonsten wird mit dem höchsten Trumpf eingestochen, sofern der Stich bisher nicht vom Nullspieler angeführt wird. In diesem Fall wird eine Karte abgeworfen oder, falls der Nullspieler eingestochen hat, ein niedrigerer Trumpf gespielt.

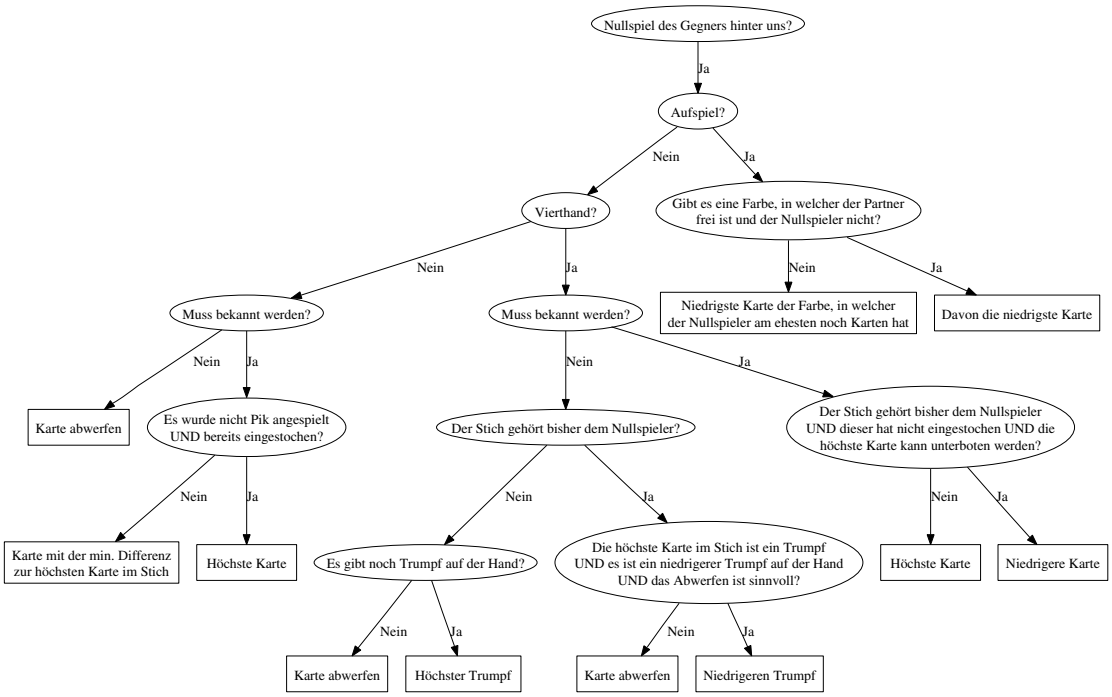

Abbildung 3.6: Statischer Entscheidungsbaum für den Gegner vor dem Nullspieler

3.4.7 Generelle Verbesserungsmöglichkeiten

Die soeben vorgestellte KI spielt (wie schon erwähnt) auf der Grundlage von verallgemeinerten Annahmen mit dem Ziel, möglichst viele Stiche zu erhalten (abgesehen vom Nullspiel). Dies ist für die Entwicklung einer Bietmethode hilfreich, für ein optimales Spiel jedoch nicht. Aus diesem Grund habe ich bei der Bestimmung der Spielart bereits eine Optimierung hinzugefügt (siehe 3.4.2, letzter Entscheidungsknoten). Obwohl bereits dadurch die Spielstärke der KI deutlich angestiegen ist (knapp 10 Prozentpunkte im Vergleich zum ursprünglichen Spieler), handelt es sich um eine sehr einfache Modifikation.

An dieser Stelle würde sich eine Verfeinerung der Spielweise anbieten. Anstatt der durchgeführten, radikalen Zieländerung könnte der Spieler bereits weitaus früher beginnen abzuschätzen, wieviele Stiche er noch erhalten wird und gegebenenfalls bereits hohe Karten abwerfen, bevor das eigene Gebot erfüllt ist. Natürlich besteht dadurch auch die Gefahr, auf Grund falscher Annahmen zu wenige Stiche zu bekommen, weil zu voreilig abgeworfen wurde.

Der größte Schwachpunkt des entwickelten Spielers stellt jedoch das Nullspiel dar, genauer gesagt die jeweiligen Gegner des Nullspielers. Diese versuchen mit höchster Priorität den Nullspieler an der Zielerreichung zu hindern, verfehlen dadurch jedoch zu oft ihre eigene Ansage. Für diesen Zielkonflikt könnte ein guter Kompromiss gesucht werden, wodurch die Spielleistung insgesamt sicherlich deutlich profitieren würde.

Spades ist ein relativ einfaches Spiel, was durch den Einsatz einer heuristischen KI bereits vernünftige Entscheidungen treffen kann; trotzdem wäre die Entwicklung einer KI auf suchbasierten Methoden durchaus interessant. Anbieten würden sich hierfür Techniken, wie sie bereits bei Bridgeprogrammen verwendet werden, z.B. eine Alpha-Beta-Suche über einen per Monte-Carlo-Simulation ergänzten Suchbaum, ähnlich der Vorgehensweise von GIB. Damit wäre es auch möglich, neue Spielstrategien zu finden bzw. die bei diesem Spieler getroffenen Annahmen zu überprüfen. Denn genau hier liegt das größte Problem aller rein heuristischer Programme: Sie können keine neuen Strategien und Vorgehensweisen ermitteln.

Ein Spadesspieler, der sowohl über eine lernfähige KI als auch eine ebensolche Bietmethode verfügt, sollte letztendlich auch für Menschen ein würdiger Gegner sein.

3.5 Bietphase

In dieser ersten Phase direkt nach dem Austeilen der Spielkarten muss die KI eine Einschätzung darüber abgeben, wieviele Stiche mit den aktuellen Karten erreicht werden können. Neben den eigenen Karten stehen als Information auch noch die Ansagen der Mitspieler zur Verfügung. Wie jedoch schon in 2.1.2 angemerkt, sind diese Daten nicht unbedingt so verlässlich und konstant, als dass eine Einbeziehung dieser in den Entscheidungsprozess unbedingt sinnvoll wäre. Die allgemeine Spielerfahrung hat gezeigt, dass selbst ungewöhnlich niedrige oder hohe Ansagen der Mitspieler das eigene Ergebnis nicht so signifikant verändern, als dass eine Berücksichtigung dieser Tatsache das Spielergebnis sonderlich beeinflussen würde.

Eine feste Modifizierung der eigenen Ansage durch die bisher genannten Stiche birgt

zudem die Gefahr der Ausnutzung durch die Gegner (siehe 2.1.7). Selbige könnten sich diese Tatsache derart zu Nutze machen, um durch geschicktes über- oder unterbieten den Spieler in Bedrängnis zu bringen, sei es durch das Erhalten von Strafpunkten oder das Nicht-Erreichen der vorhergesagten Stiche. Vor allem aus diesem Grund werden zur Stichansage nur die eigenen Karten verwendet.

Eine Besonderheit bei allen Bietmethoden stellt das Nullspiel dar. Auf Grund der besonders hohen Wertung dieses Sonderspiels mit 100 Punkten verschiebt es die Prioritäten bei der Spielweise der Gegner (und auch des Partners). Vordergründig ist nun nicht mehr das Erreichen der angesagten eigenen Stiche, sondern die Vereitlung eines Siegs der Nullpartei. Deshalb ist ein Kartenblatt, mit welchem in einem normalen Spiel wahrscheinlich nur null Stiche erreicht werden, auf keinen Fall mit einem Blatt gleichzusetzen, das für ein aktives Nullspiel geeignet ist. Die Bietmethode muss deshalb gesondert entscheiden, ob das vorliegende Blatt robust genug für ein solches Spiel ist, oder ob nicht lieber ein normales Spiel mit wenigen angesagten Stichen (meist einer) vorzuziehen ist.

Bei der Entscheidung, bei welcher Siegeswahrscheinlichkeit ein Null gespielt werden sollte, hat sich das Spielen ab einer Chance von 50 % als ideal herausgestellt. Zwar werden bei höheren Werten mehr der tatsächlich angesagten Nullspiele gewonnen, jedoch sinkt die absolute Anzahl der ausgeführten Spiele. Dadurch ist jedoch auch die Wahrscheinlichkeit auf einen Gesamtsieg vermindert.

Bezüglich der Gewinnwahrscheinlichkeit muss bei Spades keine Gegenrechnung mit anderen möglichen Spielarten erfolgen, wie dies z.B. bei Skat notwendig ist, da die Punktedifferenz im Normalfall relativ konstant ist. Durchschnittlich werden mit einem Nullblatt bei einem normalen Spiel 1,8 Stiche gewonnen. Bei einem Gebot von einem oder zwei Stichen entspräche dies 10 oder 20 Punkten, wohingegen ein gewonnenes Null immer eine Wertigkeit von 100 Punkten besitzt. Beim bereits angesprochenen Skat ist die Situation jedoch verschieden: Ein normales Nullspiel ist dort 23 Punkte wert und entspricht somit nahezu einem einfachen Kreuz mit 24 Punkten. In diesem Beispiel entscheidet demnach nur die absolute Gewinnwahrscheinlichkeit beider Spiele über das konkrete Spiel, nicht jedoch die damit zu erreichenden Punkte. Allerdings kann ein Nullspiel beim Skat auch erheblich weniger als ein Farbspiel wert sein, seltener auch umgekehrt. Deshalb muss dann eine Abschätzung der Siegeschance gegenüber der zu erreichenden Punkte stattfinden. Diese Betrachtung kann bei Spades jedoch wegfallen, weshalb ein Null grundsätzlich immer einem normalem Spiel bevorzugt wird, sofern die Aussicht auf einen Sieg groß genug ist.

3.5.1 Bestimmung der Qualität des Kartenblatts

Eines der größten Probleme bei Spades ist eine möglichst effiziente Überprüfung der Güte der eigenen Karten. Nur mit einer solchen Einschätzung ist es möglich, annähernd die richtige Stichanzahl anzusagen. Gute menschliche Spieler treffen die Entscheidung auf Grund ihrer Erfahrung. Dies ist Anfängern und vor allem Computerspielern (anfangs) nicht möglich, weshalb das Bestimmen besonderer Merkmale unumgänglich ist.

Entscheidend ist neben der reinen Anzahl der Karten einer bestimmten Farbe auch deren Wertigkeit und Verteilung. Als einfaches Beispiel sei ein blankes Ass genannt.

Da sich die anderen 12 Karten auf 3 Spieler verteilen ist die Wahrscheinlichkeit mit dieser Karte einen Stich zu machen relativ hoch. Somit bietet es sich an, darauf auch einen Stich anzusagen. Mit zunehmender Anzahl der Beikarten erhöht sich jedoch die Wahrscheinlichkeit, dass ein anderer Spieler in dieser Farbe frei ist und einstechen kann. Ist es bei diesem einfachen Beispiel noch sehr leicht, eine vernünftige Grenze zu ermitteln, ist dies bei anderen Konstellationen schon deutlich schwieriger.

Hier soll als Beispiel das Fehlen einer Farbe zu Beginn des Spieles dienen. Wird darauf ein Stich angesetzt ist es durchaus möglich, dass der Partner auf eine hohe Karte dieser Farbe ebenfalls einen Stich ansagt. Da im Team auf den gleichen Faktor zwei Stiche angesagt wurden, kann dies zu Problemen beim Erreichen der minimalen Stiche führen.

3.5.2 Statische Bietmethode

Die verwendete statische Bietmethode ist aus der ersten Überlegung nach den wesentlichen Merkmalen der Karten für das Erhalten von Stichen entstanden und wurde durch Experimente mit anderen Versionen weiter verbessert. Trotz ihrer Einfachheit ist die Fehlerrate in einem akzeptablen Bereich (hierzu mehr im Kapitel 4.5).

Die Methode geht von je einem gewonnenem Stich aus für

- Ass und König in Kreuz, Herz und Karo,
- Ass, König, Dame und Bube in Pik sowie
- die Anzahl der restlichen Pikkarten (10 bis 2) abzüglich drei (mindestens jedoch Null).

Wie leicht zu erkennen ist, werden in einem Spiel mit vier Spielern dieser Bietmethode mindestens 10 und maximal 16 Stiche angesagt. Der letzte Fall tritt jedoch nur ein, wenn ein Spieler alle Pikkarten unterhalb des Buben erhält. Im Durchschnitt liefert diese Methode in der Summe eine Vorhersage, die Nahe an den in [23] angegebenen 11 Stichen liegt.

Außer der fixen Anzahl bestimmter Karten könnten natürlich noch weitere Eigenschaften der gegebenen Karten wie z.B. das Fehlen einer Farbe berücksichtigt werden. Es hat sich jedoch herausgestellt, dass die Bietleistung der Methode dadurch (wie auch erwartet) im Allgemeinen nicht gesteigert werden kann.

Nullspiel

Wie bereits weiter oben in diesem Kapitel beschrieben, muss ein Kartenblatt für ein Nullspiel relativ robust sein, da die Gegenspieler vordergründig am Scheitern des Nullspielers arbeiten werden, wofür sie oftmals sogar die Unterschreitung der eigenen Ansagen in Kauf nehmen. Ein Nullblatt sollte deshalb nach besonderen Kriterien ausgewählt werden, im Vordergrund steht die Siegeswahrscheinlichkeit. Allerdings gibt es ein weiteres, konträres Ziel: Da Null sehr hoch bepunktet wird, sollen möglichst viele Nullspiele gespielt werden. Es ist also unumgänglich, einen Mittelweg aus Häufigkeit und Siegesgewissheit zu finden, um effizient zu spielen. Da der Partner, sofern er kann, den Nullspieler unterstützt,

kann auch mit einem nicht perfekten Blatt Null gespielt werden. Die Anforderungen sind dadurch deutlich niedriger als z.B. bei Skat.

Mit der normalen Bietmethode werden auch Nullspiele angesagt; dies geschieht genau dann, wenn in jeder Farbe weder Ass noch König vorhanden ist sowie weder Pikdame noch Pikbube. Die Siegesquote liegt mit ungefähr 66 % in einem akzeptablen Bereich, jedoch wird nur bei ca. 3 % aller Karten ein Null angesagt; ideal wäre aus Erfahrung ein Wert grob um die 10 %. Aus diesem Grund muss die Bietmethode um eine dedizierte Nullansagefunktion ergänzt werden.

Als erstes muss betrachtet werden, welche Karten für den Nullspieler am gefährlichsten sind. Vereinfacht gesagt sind dies schlichtweg alle Karten, mit denen sehr wahrscheinlich ein Stich gemacht wird, ohne dass der Nullspieler dies verhindern könnte. Dies sind vor allem

1. eine oder mehrere hohe Trumpfkarten (Ass bis Bube),
2. eine hohe Anzahl von Trumpfkarten und
3. hohe Farbkarten ohne korrespondierende Luschen (in ausreichender Anzahl).

Für diese drei Parameter müssen nun Grenzwerte gefunden, in welchen ein Null riskiert werden kann. Natürlich kann bei einer relativ groben Heuristik keine optimale Lösung für jedes mögliche Blatt gefunden werden, jedoch ist dies auch nicht unbedingt notwendig. Die statische Nullbietfunktion sagt genau dann Null an, wenn alle folgenden Bedingungen erfüllt sind:

1. Es ist keine Trumpfkarte über der Zehn auf der Hand und
2. es dürfen insgesamt nur maximal drei Trumpfkarten vorhanden sein.
3. Der Durchschnittswert der vier niedrigsten Karten darf in jeder Farbe maximal acht betragen.

Für die Errechnung des Durchschnittswerts werden die Karten aufsteigend bewertet (Zwei = 1 bis Ass = 13), die Werte der vier niedrigsten Karten (sofern vorhanden) aufaddiert und durch vier bzw. deren Anzahl geteilt. Ist in dieser Farbe keine Karte vorhanden, wird der Wert mit null definiert. Der Gedanke dabei ist, dass es bei vier niedrigen Karten in einer Farbe egal ist, was in dieser Farbe ansonsten auf der Hand ist. Würde der Durchschnitt jedoch über alle Karten einer Farbe berechnet, entfielen dadurch brauchbare Nullspiele.

Die eben genannten Werte habe ich empirisch überprüft und sie haben sich als sinnvoll herausgestellt. Der Durchschnittswert kann auch nur über die drei niedrigsten Karten ohne große Beeinflussung berechnet werden, in diesem Fall muss der Grenzwert etwas nach unten korrigiert werden.

Die eben beschriebene Funktion ergänzt nur die statische Bietmethode, ein ursprünglicher Nullwert wird weiterhin als solcher angesagt. Eine Ausnahme existiert hingegen

noch: Sollte der Partner bereits ein Null angesagt haben, wird die Bietmethode kein weiteres Null ansagen, da zwei Nullspiele in einem Team so gut wie nie gewonnen werden.

Eine weitere mögliche Einschränkung wäre der Verzicht auf ein Nullgebot, sofern zum Sieg nur noch sehr wenige Punkte fehlen, welche mit einem normalen, niedrigen Gebot sicherer zu erreichen wären. Es hat sich jedoch herausgestellt, dass durch diese Einstellung keine Veränderung der Spielleistung erreicht wird. Aus diesem Grund habe ich auch bei den lernfähigen Nullfunktionen darauf verzichtet.

Trotz ihrer Einfachheit steht diese Bietmethode den Ergebnissen der (noch folgenden) lernfähigen Funktionen nicht eklatant nach.

3.6 Spielstärke im Vergleich

Nach dem theoretischen Entwurf des Nullspielers muss dieser gegen einen bereits vorhandenen Gegner getestet werden. Der bereits erwähnte Spadesspieler des GGZ-Projekts bietet sich für diesen Zweck an. Für den Testlauf habe ich ein Team dieses neuen Spielers gegen ein Team der GGZ-KI 10.000 Runden spielen lassen. Die Ergebnisse zeigt Tabelle 3.1.

Trotz der relativ einfachen Konstruktion durch statische Entscheidungsbäume und auch der eher unkomplizierten Bietmethode ist der entstandene Spieler der GGZ-KI durchaus ebenbürtig bzw. sogar leicht überlegen. Hierzu muss noch angemerkt werden, dass dieses Ergebnis erst durch die in 3.4.2 eingeführte Optimierung erreicht wird, ohne dieselbige behält der GGZ-Spieler die Oberhand.

Vorgestellter Spieler		**GGZ-Spieler**	
Siege	**Bietabweichung S1/S3**	**Siege**	**Bietabweichung S2/S4**
53,54 %	0,59 (1,01) / 0,62 (1,17)	46,46 %	0,12 (1,26) / 0,06 (1,24)

Tabelle 3.1: Spielergebnisse gegen den GGZ-Spieler (ohne Nullspiel) mit vorzeichenbehafteter und betragsmäßiger Addition der Bietabweichung

4 Bieten nach Erfahrungswerten

Der erste Ansatz zur Verbesserung der Bietleistung eines Spadesspielers ist die Verwendung von Erfahrungswerten über die gewonnenen Stiche aus vergangenen Spielen zur Berechnung eines Gebots. In diesem Kapitel wird eine einfache und dabei trotzdem sehr wirkungsvolle Methode beschrieben, um die aus Trainingsspielen gesammelten Daten in einer effektiven Form zu speichern, auszuwerten und die Bietleistung des Spielers dadurch zu erhöhen.

Der erste Schritt ist die Entwicklung einer geeigneten Kodierung und Archivierung der verfügbaren Informationen. Als nächstes muss ein Verfahren gefunden werden, um aus den gespeicherten Werten ein Gebot für das aktuelle Spiel zu generieren, wobei die Maximierung der Punktezahl im Vordergrund steht. Abgeschlossen wird das Kapitel durch umfangreiche Testreihen gegen den statischen Spieler und die KI des GGZ-Projekts.

4.1 Reduzierung der Spielkarten zu Äquivalenzklassen

Die grundlegenste Voraussetzung für ein gutes Bietergebnis dieser Methode ist eine sinnvolle Zusammenfassung bestimmter Kartenblätter zu Äquivalenzklassen. Dieser Schritt ist notwendig, da auf Grund der immens großen Anzahl aller möglichen Kartenverteilungen eines Spielers nicht jedes Blatt einzeln bewertet werden kann. Die Anzahl der möglichen Kartenblätter eines Spielers bei Spades mit 13 aus 52 Karten ist einfach zu berechnen:

$$\binom{52}{13} = \frac{52!}{13!(52-13)!} = 635.013.559.600$$

Das erste Ziel ist damit die Entwicklung einer vernünftigen Vereinfachung der Handkarten des Spielers. Diese muss zwei konträre Bedingungen erfüllen, nämlich

1. die Anzahl der Karten möglichst stark reduzieren und dabei
2. die wesentlichen Merkmale erhalten.

Im Folgenden wird nun eine Funktion entwickelt, welche unter den gegebenen Bedingungen eine vernünftige Vereinfachung ermöglicht.

4.1.1 Mögliche Vereinfachungen

Der Algorithmus benötigt als Eingabe eine vernünftige Repräsentation der Daten, welche auf der einen Seite nicht bereits wichtige Merkmale eliminiert hat und trotzdem die

Anzahl aller Kartenblätter auf ein verarbeitbares Maß reduziert. Hierzu folgende einfache Überlegungen:

1. Die drei Farben Kreuz, Herz und Karo sind gleichwertig. Bei der Bewertung der Kartenqualität sind die Farben dadurch beliebig austauschbar. Permutationen können deshalb gemeinsam betrachtet werden.
2. Von jeder Farbe sind 13 Karten im Spiel.
3. Bei einer idealen und damit gleichmäßigen Verteilung besitzen drei Spieler je drei Karten einer bestimmten Farbe und ein Spieler vier Karten.
4. In jedem Fall sind nur maximal drei normale Farbstiche möglich bis der erste Spieler in dieser Farbe frei ist.
5. Farbstiche werden im Normalfall mit der jeweils höchsten Karte gemacht. Bei drei Stichen wären dies Ass, König und Dame.

Wie direkt zu erkennen ist, kann durch die Ausnutzung der Farbsymmetrien zur Bildung von Äquivalenzklassen die Anzahl der möglichen Kartenblätter bereits drastisch reduziert werden. Da für die Bietphase nur die jeweils hochwertigen Karten einer Farbe entscheidend sind, können insgesamt folgende Vereinfachungen durchgeführt werden:

1. Für jede Farbe wird das Vorhandensein von Ass, König und Dame sowie die Anzahl der jeweiligen Restkarten kodiert.
2. Kreuz, Herz und Karo werden anhand dieser Kodierung nach ihrer Pseudowertigkeit sortiert.
3. Das Blatt wird fortan durch den Zusammenschluss aus Pik gefolgt von den drei sortierten Farben repräsentiert.

Durch die Sortierung der Farben ist sichergestellt, dass reine Permutationen als identisches Blatt erkannt werden. Pik darf selbstverständlich nicht mitsortiert werden. Wegen der verlustbehafteten Kompression der Variationen ist ein exaktes Wiederherstellen der ursprünglichen Karten (insbesondere eine Zuordnung der einzelnen Farben) nicht mehr möglich, dies wird jedoch auch nicht benötigt.

Die Grundlage der Sortierung ist die Pseudowertigkeit der jeweiligen Farbe. Dabei kann dieser Begriff unterschiedlich aufgefasst und umgesetzt werden, ohne das Ergebnis besonders stark zu verändern. Die zwei einfachsten und dabei sehr effizienten Möglichkeiten sind die Verwendung der absoluten Anzahl der Karten einer Farbe oder die Addition der Kartenwerte. Im ersten Fall wird dementsprechend nach der Kartenzahl sortiert, im zweiten Fall nach der Kartenhöhe. Auf Grund der effizienteren Implementierung und vor allem der besseren Kompression (d.h. Bildung von weniger Äquivalenzklassen) habe ich mich für die zweite Möglichkeit entschieden.

Durch die Anwendung aller eben gezeigten Möglichkeiten zur Vereinfachung (bei der genauen Beachtung von Ass, König und Dame) werden die ursprünglich mehr als

635 Milliarden möglichen Kartenblätter eines Spielers auf 96.521 Äquivalenzklassen reduziert. Dies entspricht einem Vereinfachungsfaktor von über 6,5 Millionen.

Natürlich muss noch gezeigt werden, ob die theoretischen Grundlagen dieser Vereinfachungsform auch in der Praxis die gewünschten Resultate zeigen. Ebenso ist eine Variation der exakten Speicherung von hohen Karten möglich, z.B. nur Ass und König, Ass bis Bube oder auch eine unterschiedliche Vereinfachung für Pik und die anderen Farben.

In Tabelle 4.1 ist ausführlich vorgerechnet, zu wievielen Äquivalenzklassen alle möglichen Kartenblätter eines Spielers vereinfacht werden können. Sortiert wird in diesem Beispiel nach der Anzahl der Karten einer Farbe. In der ersten (abgetrennten) Spalte sind die möglichen Verteilungen der Karten auf Pik und die drei Farben zu sehen. Die zweite Spalte ist zur Kontrolle die Addition der verteilten Karten, hier muss immer 13 stehen. In der folgenden Spalte steht für jede Farbe die Anzahl der möglichen Verteilungen je nach Vereinfachungsform. Diese Werte stammen aus den kleinen Tabellen am rechten Seitenrand. Schließlich werden diese Werte in der vierten Spalte miteinander multipliziert und die Ergebnisse jeder Zeile aufaddiert. Unten steht schließlich für jede Vereinfachung die Gesamtanzahl der entstehenden Äquivalenzklassen. Spalte drei und vier werden für jede Vereinfachungsform wiederholt.

Der tatsächlich im Programm verwendete Kartenschlüssel funktioniert fast identisch, jedoch wird nicht nach der Anzahl der Karten, sondern nach deren Höhe sortiert. Dies führt zu etwas weniger Äquivalenzklassen, kann jedoch nicht so anschaulich vorgerechnet werden. Die Werte dieser Vereinfachung sind in Tabelle 4.3 ersichtlich, errechnet wurden sie empirisch mit einem kurzen Javaprogramm.

4.2 Verwendeter Kartenschlüssel

In 4.1.1 wurde bereits eine mögliche Vereinfachung vorgestellt. Diese wird von der lernfähigen Bietmethode benutzt, als maximaler Detailgrad werden Ass bis Bube jeder Farbe explizit gespeichert. Konkret wird ein Schlüssel für das aktuelle Blatt berechnet, welcher genau in einem 32-Bit-Register oder einer entsprechenden Ganzzahl gespeichert werden kann. Pro Farbe werden 8 Bit benötigt, die wie in Tabelle 4.2 angegeben verwendet werden.

Die ersten vier Bit stehen dabei jeweils für das Vorhandensein von Ass, König, Dame und Bube. Dabei werden diese Werte auch mit 0 belegt, wenn die entsprechende Karte zwar vorhanden, für die aktuell gewählte Vereinfachung jedoch nicht berücksichtigt wird. Die nächsten vier Bit werden für die Kodierung der Anzahl der restlichen Karten verwendet.

Wenn diese acht Bit für jede Farbe erzeugt sind, werden sie zu einem 32-Bit-Wert zusammengesetzt. An höchster Stelle steht immer Pik, die anderen Farben folgen sortiert nach ihrer Wertigkeit. Diese entspricht schlichtweg der entsprechenden Ganzzahl zwischen 0 (keine Karte der Farbe) und 249 (alle Karten dieser Farbe sind vorhanden). Selbstverständlich kann über diesen Wert keine Aussage über die Qualität der vorhandenen Karten getroffen werden. Dies ist jedoch nicht notwendig, der Wert dient ausschließlich der Sortierung.

Pik	Farbe 1	Farbe 2	Farbe 3	*Summe*				A				AK				AKD				AKDB				AKD+♣B
13	0	0	0	*13*	*1*	*1*	*1*	*1*	1	*1*	*1*	*1*	1	*1*	*1*	*1*	1	*1*	*1*	*1*	1	1	1	1	1
12	1	0	0	*13*	*2*	*2*	*1*	*1*	4	*3*	*1*	*1*	9	*4*	*1*	*1*	16	*5*	*1*	*1*	25	5	4	1	1	20
11	2	0	0	*13*	*2*	*2*	*1*	*1*	4	*4*	*1*	*1*	16	*7*	*1*	*1*	49	*11*	*1*	*1*	121	11	7	1	1	77
11	1	1	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*3*	*1*	36	*7*	*4*	*1*	112	*11*	*5*	*5*	275	11	4	4	1	176
10	3	0	0	*13*	*2*	*2*	*1*	*1*	4	*4*	*4*	*1*	16	*8*	*1*	*1*	64	*15*	*15*	*1*	225	15	8	1	1	120
10	2	1	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*3*	48	*8*	*7*	*4*	224	*15*	*11*	*5*	825	15	7	4	1	420
10	1	1	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*3*	*3*	108	*8*	*4*	*4*	512	*15*	*5*	*5*	1.875	15	4	4	4	960
9	4	0	0	*13*	*2*	*2*	*1*	*1*	4	*4*	*4*	*1*	16	*8*	*8*	*1*	64	*16*	*16*	*1*	256	16	8	1	1	128
9	3	1	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*3*	48	*8*	*8*	*4*	256	*16*	*15*	*5*	1.200	16	8	4	1	512
9	2	2	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*8*	*7*	*7*	392	*16*	*11*	*11*	1.936	16	7	7	1	784
9	2	1	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*3*	144	*8*	*7*	*4*	896	*16*	*11*	*5*	4.400	16	7	4	4	1.792
8	5	0	0	*13*	*2*	*2*	*1*	*1*	4	*4*	*4*	*1*	16	*8*	*8*	*1*	64	*16*	*16*	*1*	256	16	8	1	1	128
8	4	1	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*3*	48	*8*	*8*	*4*	256	*16*	*16*	*5*	1.280	16	8	4	1	512
8	3	2	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*8*	*8*	*7*	448	*16*	*15*	*11*	2.640	16	8	7	1	896
8	3	1	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*3*	144	*8*	*8*	*4*	1.024	*16*	*15*	*5*	6.000	16	8	4	4	2.048
8	2	2	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*3*	192	*8*	*7*	*7*	1.568	*16*	*11*	*11*	9.680	16	7	7	4	3.136
7	6	0	0	*13*	*2*	*2*	*1*	*1*	4	*4*	*4*	*1*	16	*8*	*8*	*1*	64	*16*	*16*	*1*	256	16	8	1	1	128
7	5	1	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*3*	48	*8*	*8*	*4*	256	*16*	*16*	*5*	1.280	16	8	4	1	512
7	4	2	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*8*	*8*	*7*	448	*16*	*16*	*11*	2.816	16	8	7	1	896
7	4	1	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*3*	144	*8*	*8*	*4*	1.024	*16*	*16*	*5*	6.400	16	8	4	4	2.048
7	3	3	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*8*	*8*	*8*	512	*16*	*15*	*15*	3.600	16	8	8	1	1.024
7	3	2	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	192	*8*	*8*	*7*	1.792	*16*	*15*	*11*	13.200	16	8	7	4	3.584
7	2	2	2	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*8*	*7*	*7*	2.744	*16*	*11*	*11*	21.296	16	7	7	7	5.488
6	7	0	0	*13*	*2*	*2*	*1*	*1*	4	*4*	*4*	*1*	16	*8*	*8*	*1*	64	*16*	*16*	*1*	256	16	8	1	1	128
6	6	1	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*3*	48	*8*	*8*	*4*	256	*16*	*16*	*5*	1.280	16	8	4	1	512
6	5	2	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*8*	*8*	*7*	448	*16*	*16*	*11*	2.816	16	8	7	1	896
6	5	1	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*3*	144	*8*	*8*	*4*	1.024	*16*	*16*	*5*	6.400	16	8	4	4	2.048
6	4	3	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*8*	*8*	*8*	512	*16*	*16*	*15*	3.840	16	8	8	1	1.024
6	4	2	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	192	*8*	*8*	*7*	1.792	*16*	*16*	*11*	14.080	16	8	7	4	3.584
6	3	3	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	192	*8*	*8*	*8*	2.048	*16*	*15*	*15*	18.000	16	8	8	4	4.096
6	3	2	2	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*8*	*8*	*7*	3.136	*16*	*15*	*11*	29.040	16	8	7	7	6.272
5	8	0	0	*13*	*2*	*2*	*1*	*1*	4	*4*	*4*	*1*	16	*8*	*8*	*1*	64	*16*	*16*	*1*	256	16	8	1	1	128
5	7	1	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*3*	48	*8*	*8*	*4*	256	*16*	*16*	*5*	1.280	16	8	4	1	512
5	6	2	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*8*	*8*	*7*	448	*16*	*16*	*11*	2.816	16	8	7	1	896
5	6	1	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*3*	144	*8*	*8*	*4*	1.024	*16*	*16*	*5*	6.400	16	8	4	4	2.048
5	5	3	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*8*	*8*	*8*	512	*16*	*16*	*15*	3.840	16	8	8	1	1.024
5	5	2	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	192	*8*	*8*	*7*	1.792	*16*	*16*	*11*	14.080	16	8	7	4	3.584
5	4	4	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*8*	*8*	*8*	512	*16*	*16*	*16*	4.096	16	8	8	1	1.024
5	4	3	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	192	*8*	*8*	*8*	2.048	*16*	*16*	*15*	19.200	16	8	8	4	4.096
5	4	2	2	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*8*	*8*	*7*	3.136	*16*	*16*	*11*	30.976	16	8	7	7	6.272
5	3	3	2	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*8*	*8*	*8*	3.584	*16*	*16*	*15*	39.600	16	8	8	7	7.168
4	9	0	0	*13*	*2*	*2*	*1*	*1*	4	*4*	*4*	*1*	16	*8*	*8*	*1*	64	*16*	*16*	*1*	256	16	8	1	1	128
4	8	1	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*3*	48	*8*	*8*	*4*	256	*16*	*16*	*5*	1.280	16	8	4	1	512
4	7	2	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*8*	*8*	*7*	448	*16*	*16*	*11*	2.816	16	8	7	1	896
4	7	1	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*3*	144	*8*	*8*	*4*	1.024	*16*	*16*	*5*	6.400	16	8	4	4	2.048
4	6	3	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*8*	*8*	*8*	512	*16*	*16*	*15*	3.840	16	8	8	1	1.024
4	6	2	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	192	*8*	*8*	*7*	1.792	*16*	*16*	*11*	14.080	16	8	7	4	3.584
4	5	4	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*8*	*8*	*8*	512	*16*	*16*	*16*	4.096	16	8	8	1	1.024
4	5	3	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	192	*8*	*8*	*8*	2.048	*16*	*16*	*15*	19.200	16	8	8	4	4.096
4	5	2	2	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*8*	*8*	*7*	3.136	*16*	*16*	*11*	30.976	16	8	7	7	6.272
4	4	4	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*8*	*8*	*8*	4.096	*16*	*16*	*16*	65.536	16	8	8	8	8.192
4	4	3	2	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*8*	*8*	*8*	4.096	*16*	*16*	*15*	54.000	16	8	8	8	8.192
4	3	3	3	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*8*	*8*	*8*	4.096	*16*	*15*	*15*	54.000	16	8	8	8	8.192
3	10	0	0	*13*	*2*	*2*	*1*	*1*	4	*4*	*4*	*1*	16	*8*	*8*	*1*	64	*15*	*15*	*1*	225	15	8	1	1	120
3	9	1	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*3*	48	*8*	*8*	*4*	256	*15*	*16*	*5*	1.200	15	8	4	1	480
3	8	2	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*8*	*8*	*7*	448	*15*	*16*	*11*	2.640	15	8	7	1	840
3	8	1	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*3*	144	*8*	*8*	*4*	1.024	*15*	*16*	*5*	6.000	15	8	4	4	1.920
3	7	3	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*8*	*8*	*8*	512	*15*	*16*	*15*	3.600	15	8	8	1	960
3	7	2	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	192	*8*	*8*	*7*	1.792	*15*	*16*	*11*	13.200	15	8	7	4	3.360
3	6	4	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*8*	*8*	*8*	512	*15*	*16*	*16*	3.840	15	8	8	1	960
3	6	3	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	192	*8*	*8*	*8*	2.048	*15*	*16*	*15*	18.000	15	8	8	4	3.840
3	6	2	2	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*8*	*8*	*7*	3.136	*15*	*16*	*11*	29.040	15	8	7	7	5.880
3	5	5	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*8*	*8*	*8*	512	*15*	*16*	*16*	3.840	15	8	8	1	960
3	5	4	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	192	*8*	*8*	*8*	2.048	*15*	*16*	*16*	19.200	15	8	8	4	3.840
3	5	3	2	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*8*	*8*	*8*	3.584	*15*	*16*	*15*	39.600	15	8	8	7	6.720
3	4	4	2	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*8*	*8*	*8*	3.584	*15*	*16*	*16*	42.240	15	8	8	7	6.720
3	4	3	3	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*8*	*8*	*8*	4.096	*15*	*16*	*15*	54.000	15	8	8	8	7.680
2	11	0	0	*13*	*2*	*2*	*1*	*1*	4	*4*	*4*	*1*	16	*7*	*7*	*1*	49	*11*	*11*	*1*	121	11	7	1	1	77
2	10	1	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*3*	48	*7*	*8*	*4*	224	*11*	*15*	*5*	825	11	8	4	1	352
2	9	2	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*7*	*8*	*7*	392	*11*	*16*	*11*	1.936	11	8	7	1	616
2	9	1	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*3*	144	*7*	*8*	*4*	896	*11*	*16*	*5*	4.400	11	8	4	4	1.408
2	8	3	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*7*	*8*	*8*	448	*11*	*16*	*15*	2.640	11	8	8	1	704
2	8	2	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*3*	192	*7*	*8*	*7*	1.568	*11*	*16*	*11*	9.680	11	8	7	4	2.464
2	7	4	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*7*	*8*	*8*	448	*11*	*16*	*16*	2.816	11	8	8	1	704
2	7	3	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	192	*7*	*8*	*8*	1.792	*11*	*16*	*15*	13.200	11	8	8	4	2.816
2	7	2	2	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*7*	*8*	*7*	2.744	*11*	*16*	*11*	21.296	11	8	7	7	4.312
2	6	5	0	*13*	*2*	*2*	*2*	*1*	8	*4*	*4*	*4*	64	*7*	*8*	*8*	448	*11*	*16*	*16*	2.816	11	8	8	1	704
2	6	4	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	192	*7*	*8*	*8*	1.792	*11*	*16*	*16*	14.080	11	8	8	4	2.816
2	6	3	2	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*7*	*8*	*8*	3.136	*11*	*16*	*15*	29.040	11	8	8	7	4.928
2	5	5	1	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	192	*7*	*8*	*8*	1.792	*11*	*16*	*16*	14.080	11	8	8	4	2.816
2	5	4	2	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*7*	*8*	*8*	3.136	*11*	*16*	*16*	30.976	11	8	8	7	4.928
2	5	3	3	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*7*	*8*	*8*	3.584	*11*	*16*	*15*	39.600	11	8	8	8	5.632
2	4	4	3	*13*	*2*	*2*	*2*	*2*	16	*4*	*4*	*4*	256	*7*	*8*	*8*	3.584	*11*	*16*	*16*	42.240	11	8	8	8	5.632
1	12	0	0	*13*	*2*	*2*	*1*	*1*	4	*3*	*3*	*1*	9	*4*	*4*	*1*	16	*5*	*5*	*1*	25	5	4	1	1	20
1	11	1	0	*13*	*2*	*2*	*2*	*1*	8	*3*	*4*	*3*	36	*4*	*7*	*4*	112	*5*	*11*	*5*	275	5	7	4	1	140
1	10	2	0	*13*	*2*	*2*	*2*	*1*	8	*3*	*4*	*4*	48	*4*	*8*	*7*	224	*5*	*15*	*11*	825	5	8	7	1	280
1	10	1	1	*13*	*2*	*2*	*2*	*2*	16	*3*	*4*	*3*	108	*4*	*8*	*4*	512	*5*	*15*	*5*	1.875	5	8	4	4	640
1	9	3	0	*13*	*2*	*2*	*2*	*1*	8	*3*	*4*	*4*	48	*4*	*8*	*8*	256	*5*	*16*	*15*	1.200	5	8	8	1	320
1	9	2	1	*13*	*2*	*2*	*2*	*2*	16	*3*	*4*	*4*	144	*4*	*8*	*7*	896	*5*	*16*	*11*	4.400	5	8	7	4	1.120
1	8	4	0	*13*	*2*	*2*	*2*	*1*	8	*3*	*4*	*4*	48	*4*	*8*	*8*	256	*5*	*16*	*16*	1.280	5	8	8	1	320
1	8	3	1	*13*	*2*	*2*	*2*	*2*	16	*3*	*4*	*4*	144	*4*	*8*	*8*	1.024	*5*	*16*	*15*	6.000	5	8	8	4	1.280
1	8	2	2	*13*	*2*	*2*	*2*	*2*	16	*3*	*4*	*4*	192	*4*	*8*	*7*	1.568	*5*	*16*	*11*	9.680	5	8	7	7	1.960
1	7	5	0	*13*	*2*	*2*	*2*	*1*	8	*3*	*4*	*4*	48	*4*	*8*	*8*	256	*5*	*16*	*16*	1.280	5	8	8	1	320
1	7	4	1	*13*	*2*	*2*	*2*	*2*	16	*3*	*4*	*4*	144	*4*	*8*	*8*	1.024	*5*	*16*	*16*	6.400	5	8	8	4	1.280
1	7	3	2	*13*	*2*	*2*	*2*	*2*	16	*3*	*4*	*4*	192	*4*	*8*	*8*	1.792	*5*	*16*	*15*	13.200	5	8	8	4	2.240
1	6	6	0	*13*	*2*	*2*	*1*	*1*	4	*3*	*4*	*4*	48	*4*	*8*	*8*	256	*5*	*16*	*16*	1.280	5	8	8	1	320
1	6	5	1	*13*	*2*	*2*	*2*	*2*	16	*3*	*4*	*4*	144	*4*	*8*	*8*	1.024	*5*	*16*	*16*	6.400	5	8	8	4	1.280
1	6	4	2	*13*	*2*	*2*	*2*	*2*	16	*3*	*4*	*4*	192	*4*	*8*	*8*	1.792	*5*	*16*	*16*	14.080	5	8	8	7	2.240
1	6	3	3	*13*	*2*	*2*	*2*	*2*	16	*3*	*4*	*4*	192	*4*	*8*	*8*	2.048	*5*	*16*	*15*	18.000	5	8	8	8	2.560
1	5	5	2	*13*	*2*	*2*	*2*	*2*	16	*3*	*4*	*4*	192	*4*	*8*	*8*	1.792	*5*	*16*	*16*	14.080	5	8	8	7	2.240
1	5	4	3	*13*	*2*	*2*	*2*	*2*	16	*3*	*4*	*4*	192	*4*	*8*	*8*	2.048	*5*	*16*	*16*	19.200	5	8	8	8	2.560
1	4	4	4	*13*	*2*	*2*	*2*	*2*	16	*3*	*4*	*4*	192	*4*	*8*	*8*	2.048	*5*	*16*	*16*	20.480	5	8	8	8	2.560
0	13	0	0	*13*	*1*	*1*	*1*	*1*	1	*1*	*1*	*1*	1	*1*	*1*	*1*	1	*1*	*1*	*1*	1	1	1	1	1	1
0	12	1	0	*13*	*1*	*2*	*2*	*1*	4	*1*	*3*	*1*	9	*1*	*4*	*4*	16	*1*	*5*	*5*	25	1	4	4	1	16
0	11	2	0	*13*	*1*	*2*	*2*	*1*	4	*1*	*4*	*4*	16	*1*	*7*	*7*	49	*1*	*11*	*11*	121	1	7	7	1	49
0	11	1	1	*13*	*1*	*2*	*2*	*2*	8	*1*	*4*	*3*	36	*1*	*7*	*4*	112	*1*	*11*	*5*	275	1	7	4	4	112
0	10	3	0	*13*	*1*	*2*	*2*	*1*	4	*1*	*4*	*4*	16	*1*	*8*	*8*	64	*1*	*15*	*15*	225	1	8	8	1	64
0	10	2	1	*13*	*1*	*2*	*2*	*2*	8	*1*	*4*	*4*	48	*1*	*8*	*7*	224	*1*	*15*	*11*	825	1	8	7	4	224
0	9	4	0	*13*	*1*	*2*	*2*	*1*	4	*1*	*4*	*4*	16	*1*	*8*	*8*	64	*1*	*16*	*16*	256	1	8	8	1	64
0	9	3	1	*13*	*1*	*2*	*2*	*2*	8	*1*	*4*	*3*	48	*1*	*8*	*4*	256	*1*	*16*	*5*	1.200	1	8	4	4	256
0	9	2	2	*13*	*1*	*2*	*2*	*2*	8	*1*	*4*	*4*	64	*1*	*8*	*7*	392	*1*	*16*	*11*	1.936	1	8	7	7	392
0	8	5	0	*13*	*1*	*2*	*2*	*1*	4	*1*	*4*	*4*	16	*1*	*8*	*8*	64	*1*	*16*	*16*	256	1	8	8	1	64
0	8	4	1	*13*	*1*	*2*	*2*	*2*	8	*1*	*4*	*3*	48	*1*	*8*	*8*	256	*1*	*16*	*16*	1.280	1	8	8	4	256
0	8	3	2	*13*	*1*	*2*	*2*	*2*	8	*1*	*4*	*4*	64	*1*	*8*	*8*	448	*1*	*16*	*15*	2.640	1	8	8	7	448
0	7	6	0	*13*	*1*	*2*	*2*	*1*	4	*1*	*4*	*4*	16	*1*	*8*	*8*	64	*1*	*16*	*16*	256	1	8	8	1	64
0	7	5	1	*13*	*1*	*2*	*2*	*2*	8	*1*	*4*	*3*	48	*1*	*8*	*8*	256	*1*	*16*	*16*	1.280	1	8	8	4	256
0	7	4	2	*13*	*1*	*2*	*2*	*2*	8	*1*	*4*	*4*	64	*1*	*8*	*8*	448	*1*	*16*	*16*	2.816	1	8	8	7	448
0	7	3	3	*13*	*1*	*2*	*2*	*2*	8	*1*	*4*	*4*	64	*1*	*8*	*8*	512	*1*	*16*	*15*	3.600	1	8	8	8	512
0	6	6	1	*13*	*1*	*2*	*2*	*2*	8	*1*	*4*	*3*	48	*1*	*8*	*8*	256	*1*	*16*	*16*	1.280	1	8	8	4	256
0	6	5	2	*13*	*1*	*2*	*2*	*2*	8	*1*	*4*	*4*	64	*1*	*8*	*8*	448	*1*	*16*	*16*	2.816	1	8	8	7	448
0	6	4	3	*13*	*1*	*2*	*2*	*2*	8	*1*	*4*	*4*	64	*1*	*8*	*8*	512	*1*	*16*	*16*	3.840	1	8	8	8	512
0	5	5	3	*13*	*1*	*2*	*2*	*2*	8	*1*	*4*	*4*	64	*1*	*8*	*8*	512	*1*	*16*	*16*	3.840	1	8	8	8	512
0	5	4	4	*13*	*1*	*2*	*2*	*2*	8	*1*	*4*	*4*	64	*1*	*8*	*8*	512	*1*	*16*	*16*	4.096	1	8	8	8	512
								A				**AK**				**AKD**				**AKDB**				**AKD+♣B**
								1.322				13.537				129.021				1.116.958				224.077

Kartenzahl	Möglichkeiten A
13	1
12	2
11	2
10	2
9	2
8	2
7	2
6	2
5	2
4	2
3	2
2	2
1	2
0	1

Kartenzahl	Möglichkeiten AK
13	1
12	3
11	4
10	4
9	4
8	4
7	4
6	4
5	4
4	4
3	4
2	4
1	3
0	1

Kartenzahl	Möglichkeiten AKD
13	1
12	4
11	7
10	8
9	8
8	8
7	8
6	8
5	8
4	8
3	8
2	7
1	4
0	1

Kartenzahl	Möglichkeiten AKDB
13	1
12	5
11	11
10	15
9	16
8	16
7	16
6	16
5	16
4	16
3	15
2	11
1	5
0	1

Tabelle 4.1: Berechnung der Äquivalenzklassen pro Vereinfachungsstufe

Bit	8	7	6	5	4	3	2	1
Datum	Ass	König	Dame	Bube	*Anzahl*	*Anzahl*	*Anzahl*	*Anzahl*

Tabelle 4.2: Bitweise Speicherung der vereinfachten Karten einer Farbe

Nicht optimal ist der benötigte Speicherplatz des Verfahrens. Da die Information jedoch in einem 32-Bit-Integer gespeichert wird, brächte eine platzoptimierte Variante keine reale Einsparung. Ein Vorteil daran ist, dass die Werte sehr effizient berechnet werden können und zudem das Ergebnis in einer für Menschen lesbaren Form vorliegt.

4.2.1 Speicherung

Zur Speicherung der Erfahrungswerte bietet sich eine Hashtabelle oder ähnliche Datenstrukturen an. In diesen kann der errechnete Kartenschlüssel direkt als Schlüssel dienen und auf ein Objekt verweisen, welches die notwendigen Spieldaten enthält. Die Charakteristik der hier benötigten Datenverwaltung entspricht ziemlich genau dem idealen Einsatzzweck einer Hashtabelle: Es werden nur Daten hinzugefügt und nicht gelöscht, zudem findet der Zugriff wahlfrei statt. Nach der initialen Lernphase werden an der eigentlichen Struktur nur noch wenige Veränderungen vorgenommen. Neu aufgenommene Werte werden direkt in die entsprechenden Objekte hinzugefügt, sofern der entsprechende Eintrag bereits vorhanden ist.

4.3 Trainingsphase

Das Trainieren der Bietkomponente ist sehr unproblematisch, da lediglich reelle Spielergebnisse benötigt werden. Voraussetzung ist lediglich, dass die Trainingsdaten von der gleichen KI stammen, die auch die neu trainierte Bietkomponente später nutzen soll. Im Idealfall ist zudem der Gegner aus den Trainingsspielen der Gleiche.

Für ein effizientes Training gegen die eigene Spiellogik bietet es sich an, vier KI-Spieler zu verwenden, die unter Verwendung der gleichen Datenbasis gegeneinander spielen. Dadurch werden pro Spielrunde vier Kartenblätter analysiert und die damit möglichen Erfolge erlernt. Bei einem Training gegen andere Spieler ist dies natürlich nicht möglich, in diesem Fall können lediglich die im Team spielenden KI-Spieler einen Erfahrungsspeicher teilen.

Gespeichert werden die Kartenblätter mit den zugehörigen Ergebnissen aller Spiele. Eine Ausnahme hiervon bildet nur das Nullspiel eines Spielers. Wegen der eklatant anderen Spielweise ist das hierbei entstehende Ergebnis nicht repräsentativ für ein normales Spiel mit den gleichen Karten und würde zu einer Verfälschung der Datenbasis führen. Deshalb werden nur Spielrunden ohne ein angesagtes Null für das Training verwendet. Das Nullspiel wird im Abschnitt 4.4.2 auch hinsichtlich der besonderen Trainingsanforderungen näher behandelt.

4.4 Eigentliche Bietfunktion

Nachdem mit ausreichend vielen Spielergebnissen trainiert wurde, kann die Bietkomponente mit dieser Hilfe relativ präzise Vorhersagen über die mit einem bestimmten Blatt zu erreichenden Stiche liefern. Als Ausgangsdaten stehen der Bietmethode die Informationen bereit, ob das entsprechende (vereinfachte) Kartenblatt in der Vergangenheit bereits vorkam und wenn ja, in welcher Anzahl von Spielen wieviele Stiche erreicht wurden. Der daraus errechenbare Durchschnitt sollte jedoch als Rohwert aufgefasst werden. Durch die verschiedene Bepunktung von Über- oder Unteransagen bei Spades muss dieser Wert so in ein ganzzahliges Gebot verwandelt werden, dass der maximal mögliche Gewinn am Ende des Spiels erreicht wird.

An dieser Stelle tauchen zwei Probleme auf:

1. Das auszuwertende Kartenblatt ist bisher nicht aufgetreten oder
2. die vergangenen Vorkommen reichen nicht für ein vernünftiges Gebot aus.

Im ersten Fall muss auf ein alternatives Bietsystem zurückgegriffen werden, wofür sich natürlich die statische Methode anbietet. Möglich ist jedoch auch das Zurückgreifen auf eine andere Hashtabelle (mit einer höheren Vereinfachungsstufe) oder das Verwenden eines neuronalen Netzes.

Generell steigt die Qualität der Vorhersage anfänglich sehr stark mit der Anzahl der Vorkommnisse eines bestimmten Blatts. Deshalb ist es sehr fraglich, ob bei einem geringen Auftreten (z.B. erst einmal) auf dieser Basis überhaupt ein sinnvolles Gebot erfolgen kann. Über einen Parameter kann man aus diesem Grund in der Bietmethode einen Grenzwert für das minimale Auftreten eines Blatts definieren, ab dessen Erreichen erst der gespeicherte Wert verwendet wird. Als vernünftig hat sich dabei ein Grenzwert von 10 herausgestellt. Ansonsten wird, wie im ersten Fall, auf eine alternative Berechnung des Gebots zurückgegriffen.

4.4.1 Korrektur der Erfahrungswerte

Die einfachste Möglichkeit, den erlernten Durchschnitt der erhaltenen Stiche in eine konkrete Ansage zu verwandeln, ist, den Wert echt zu runden. Vor allem auf Grund der charakteristischen Spadesbepunktung, die eine zu hohe Ansage wesentlich höher bestraft als eine zu niedrige, hat sich dies jedoch als nicht optimal herausgestellt. In der Tat ist das Spielergebnis der so behandelten Werte sogar signifikant schlechter als das der statischen Methode.

Eine weitere wichtige Rolle spielt der aktuelle Gegner. Selbstverständlich wäre es am besten, für jeden Gegner neue Erfahrungswerte zu sammeln, dies ist jedoch oftmals nicht möglich. Gerade beim Spiel gegen Menschen stellt die reine Menge der benötigten Trainingsspiele ein unüberwindbares Hindernis dar. Die Gültigkeit der erlernten Qualität der Kartenblätter ist jedoch nicht ohne Weiteres auf andere Gegner übertragbar. Ist die Spielstärke der gegnerische Mitspieler höher als die der Trainingspartner, werden weniger Stiche erreicht als durch die Daten angegeben werden, umgekehrt gilt natürlich das

Gleiche. Aus diesem Grund ist es notwendig einen Korrekturwert anzugeben, der mit dem jeweiligen Erfahrungswert verrechnet wird. Für ein Spiel gegen den gleichen Gegner hat sich eine Korrektur von -0,5 bewährt (siehe 4.5). Dies entspricht schlichtweg dem Abrunden der Durchschnittswerte.

Natürlich beeinflusst diese Korrektur direkt die Abweichung zwischen Ansage und tatsächlich gewonnenen Stichen. Ohne Bietkorrektur liegt die Abweichung durchschnittlich fast bei null, d.h. die Bietmethode sagt prinzipiell tatsächlich ziemlich genau die richtige Stichzahl voraus. Mit dem vorgeschlagenen Abzug von 0,5 auf das Gebot verschiebt sich die Abweichung, wie erwartet wird im Durchschnitt nun ein halber Stich zu wenig angesagt. Die Wahrscheinlichkeit auf eine korrekte Erfüllung des Gebots steigt dadurch stark an.

Die interessante Frage an dieser Stelle ist, ob diese Bietabweichung ideal für diese Bietfunktion ist oder sogar allgemein gilt, also ein Optimum aus dem Risiko, das Gebot nicht zu erfüllen und der höheren Wahrscheinlichkeit auf zuviele Stiche und damit verbundenen Strafpunkten darstellt. Eine Beantwortung dieser Frage ist nicht leicht, da die Berechnung eines Idealwerts nicht ohne Weiteres möglich ist. Zwar ist der Punktabzug eines zuwenig angesagten Stichs in jedem Fall gleich (insgesamt -9 Punkte), jedoch ist dies für einen zuviel angesagten Stich abhängig von der angesagten Gesamtanzahl (siehe 2.1.4). Als Hinweis können die Ergebnisse der Testläufe mit den unterschiedlichen Bietfunktionen dienen. Alle Methoden erreichen ihre Maximalleistung unabhängig vom Gegner bei einer Bietabweichung von ca. 0,5. Diese Ergebnisse lassen sich zwar nicht verallgemeinern, jedoch ist damit ein guter Wert für alle hier getesteten Methoden gefunden. Evtl. könnte die Spielleistung des GGZ-Spielers durch eine Korrektur seiner Bietfunktion auch weiter gesteigert werden.

4.4.2 Nullspiel

Obwohl das Lernverfahren für die Nullkarten prinzipiell dem bereits beschriebenen Vorgehen gleicht, gibt es einige Unterschiede. In der Einführung des Kapitels wurde erklärt, weshalb ein aktives Nullspiel andere Voraussetzungen an das Kartenblatt stellt als die normale Bewertung. Da mit den bisher gesammelten Daten kein vernünftiges Nullgebot hergeleitet werden kann, ist es unumgänglich, eine getrennte Methode für diese Bewertung zu entwickeln.

Grundsätzlich wird das bisherige Modell hierfür lediglich erweitert. Zusätzlich wird für jedes Kartenblatt nun noch gespeichert, wie oft mit selbigem ein Null gespielt und wieviele dieser Spiele auch gewonnen wurden. Um diese Daten zu erfassen bietet es sich in der Trainingsphase an, einen Spieler nur Nullspiele spielen zu lassen. Dadurch werden auch für Blätter, bei denen die statische Methode kein Null ansagen würde, verlässliche Werte ermittelt.

In der Bietphase wird, ähnlich dem bereits bekannten Vorgehen, die Anzahl der durchgeführten Nullspiele mit einem vorher festgelegten Grenzwert verglichen und zudem die Siegeswahrscheinlichkeit eines Nullspiels mit diesen Karten errechnet. Ist die Gewinnaussicht ausreichend groß (in der Regel mindestens 50 %), wird ein Nullspiel angesagt. Sollte eine dieser Bedingungen nicht zutreffen, wird mit der normalen Bietmethode fortgefahren.

Während die allgemeine Bietfunktion mit einem leeren Erfahrungsspeicher beginnen kann und bereits in der Trainingsphase als Gegner für Spades verwendet werden kann, ist dies für die Vorhersage des Nullspiels nicht ohne weiteres möglich. Dieser Unterschied resultiert aus dem Sonderspielstatus von Null: Während das Ergebnis einer normal gespielten Runde unabhängig von der verwendeten Bietmethode immer in den Erfahrungsspeicher aufgenommen werden kann, ist ein Ergebnis für Null nur verfügbar, wenn dies vorher angesagt und damit auch gespielt wurde. Daraus ergibt sich beim erstmaligen Start das Problem, dass für die gegebenen Karten mit der Bietmethode nicht vorausgesagt werden kann, ob ein Null gespielt werden soll. Tritt das aktuelle Blatt zum ersten Mal auf, kann durch die statische Methode ein Nullspiel angesagt werden. Dies führt dann jedoch zu keinem Eintrag bezüglich eines normalen Spiels für das Blatt, so dass für diese Karten immer ein Null gespielt werden würde. Umgekehrt ist der Fall ebenfalls problematisch, wenn bereits Spieldaten für die Karten vorliegen, jedoch keine Erfahrungswerte für Null. In diesem Fall müsste für ein Nullspiel die statische Nullvorhersage höher priorisiert werden als die Vorhersage der lernfähigen Funktion für ein Standardspiel. Doch selbst in diesem Fall würden nicht alle, für ein aktives Nullspiel geeigneten Karten korrekt erlernt werden. Zwar fielen die eigentlich ungeeigneten Blätter, mit denen die statische Methode ein Null spielt, heraus, jedoch würden keine Blätter erkannt werden, mit denen die statische Funktion kein Null ansagt.

Ein weiteres Hindernis schließt sich direkt an: Selbst wenn Erfahrungswerte für das Nullspiel vorliegen, es jedoch seltener gespielt wurde als der Grenzwert erfordert, wird mit diesem Blatt nie ein Null angesagt. Wird der Grenzwert hingegen nicht beachtet, sorgt ein Verlust des Nulls beim ersten Versuch für eine Gewinnwahrscheinlichkeit von 0 %, was das Blatt wiederum für alle Zeit ausschließt, auch wenn es eigentlich gut für das Sonderspiel geeignet wäre. Sehr selten auftretende Blätter und solche mit (angenommenen) schlechten Gewinnaussichten werden folglich nicht mehr verwendet; der Erfahrungswert kann nicht weiter ausgebaut werden. Aktualisiert werden nur noch vermeintlich erfolgreiche Blätter, mit welchen auch gespielt wird. Diese können dadurch jedoch durch ein oder mehrere verlorene Spiele ebenfalls in die nicht mehr aktualisierte Kategorie fallen.

Im Endeffekt kann sich bei einem Training durch gewöhnliche Spiele kein neues Blatt für ein Null qualifizieren, bereits bekannte Blätter können lediglich entfallen. Aus diesen Gründen ist es notwendig, eine dedizierte Trainingsphase mit erzwungenen Nullspielen (in denen wirklich genau ein Spieler immer Null ansagt) für möglichst viele Karten durchzuführen. Obwohl diese Vorgehensweise notwendig ist, offenbart sich gleich deren größter Nachteil: Der Trainingsaufwand gegenüber den ausschließlich normalen Spielen verfünffacht sich. Ausschlaggebend hierfür ist die Tatsache, dass exakt ein Spieler ein Null spielt, wohingegen beim normalen Training in einer Runde die Kartenblätter aller vier Spieler ausgewertet werden können. Für die gleiche Anzahl erlernter Karten ist damit die vierfache Rundenzahl *zusätzlich* zum normalen Training erforderlich.

4.5 Ergebnisse

An dieser Stelle soll nun untersucht werden, welche Steigerung der Spielleistung durch die beschriebenen Methoden erreicht werden kann. Daneben ist jedoch auch der notwendige Lernaufwand interessant, ebenso ein detaillierter Vergleich der Gebotscharakteristik mit der statischen Methode.

4.5.1 Lernaufwand

Für vernünftige Spielergebnisse ist es natürlich erheblich, wieviele Kartenblätter bisher gespielt und damit auch gespeichert wurden. Da zudem nur Blätter verwendet werden, die öfter als ein definierter Grenzwert vorkamen, spielt auch dies eine Rolle. In Tabelle 4.3 sind die insgesamt vorhandenen und die tatsächlich gespeicherten Äquivalenzklassen nach 100 Millionen ausgewerteten Kartenblättern für jede Vereinfachungsstufe angegeben. Dies entspricht 25 Millionen gespielten Runden mit vier lernfähigen Spielern, welche die gleiche Hashtabelle verwenden.

Auffällig ist der starke Anstieg der existierenden Klassen mit dem Detailgrad der Vereinfachung. Für die ersten zwei Stufen wären wesentlich weniger Spielrunden notwendig gewesen, für die Stufe *Ass bis Bube* dagegen noch weitaus mehr. Hier sind nicht einmal 50 % aller möglichen Varianten mehr als zehnmal vorgekommen.

Neben diesen absoluten Werten ist natürlich die Entwicklung der Erfahrungswerte interessant. Das Wachstum der Einträge illustriert Graph 4.1. Die charakteristische Form der einzelnen Lernkurven resultiert aus der ungleichmäßigen Verteilung der Kartenblätter. Einige Kartenklassen kommen bedeutend häufiger vor als andere, weshalb für diese bereits nach recht wenigen Spielrunden ausreichend viele Werte vorliegen. Häufig auftretende Blätter bestehen hauptsächlich aus niedrigen Farbkarten, da diese großzügig zusammengefasst werden können. Ein gegenteiliges Beispiel ist ein Blatt aus 13 Pikkarten. Dieses kommt sehr selten vor und ist der einzige Vertreter seiner Klasse.

Bedingt dadurch repräsentieren häufig auftretende Kartenklassen Blätter, mit denen nicht viele Stiche zu erwarten sind. Hochwertige Karten mit vielen Bildern erzeugen hingegen eigene Klassen, die jedoch seltener vorkommen. Dies wird von Graph 4.2 verdeutlicht. Hier sind die mit einer Kartenklasse gewonnenen Stiche bezüglich des Auftretens der Klasse aufgetragen. Die Streuung, aber vor allem auch die Höhe der erhaltenen Stiche, nimmt mit der Häufigkeit des Auftretens erheblich ab.

Nach einer ersten und recht schnellen Grundsättigung werden daher nur noch langsam neue Variationen aufgenommen. Die Spielstärke ist jedoch bereits mit einem unvollständigen Erfahrungsspeicher gegenüber der statischen Bietfunktion gesteigert, da für die gebräuchlichsten Blätter schon Daten vorliegen. Nach einer Million Spielrunden mit der Vereinfachungsstufe *Ass bis Dame* (bzw. 250.000 Runden bei vier lernenden Spielern) sind bereits ca. 52.500 Kartenklassen gespeichert. In diesem Spiel, was mit leerem Speicher gestartet wurde, konnten bereits über 72,5 % der benötigten Blätter beim Gebot durch den Erfahrungsspeicher bei einem Grenzwert von mind. zehn Vorkommen bewertet werden, beim Verzicht auf einen Grenzwert sogar knapp 95 %. Eine Steigerung der Abdeckung spielrelevanter Blätter findet bei steigendem Vollständigkeitsgrad der Hash-

tabelle nur noch sehr langsam statt und ist für ein erfolgreiches Spiel nicht unbedingt notwendig.

Alle eben genannten Werte gelten ebenso für das Nullspiel, nur dass für dieses zusätzlich mit vierfachem Aufwand trainiert werden muss (siehe 4.4.2).

Vereinfachung	ÄK insg.	ÄK	Prozent	ÄK ≥ 10	Prozent
Ass	1.162	1.115	95,96 %	1.073	92,34%
Ass und König	10.849	10.415	96,00 %	9.699	89,40 %
Ass bis Dame	96.521	90.328	93,58 %	75.759	78,49 %
Ass bis Dame + Pikbube	166.293	152.720	91,84 %	120.115	72,23 %
Ass bis Bube	791.678	666.097	84,14 %	395.488	49,96 %

Tabelle 4.3: Anzahl der existierenden und gespeicherten Äquivalenzklassen je Vereinfachungsstufe nach 100 Millionen ausgewerteten Kartenblättern

4.5.2 Spielstärke

In diesem Abschnitt wird die Steigerung der allgemeinen Spielleistung durch die lernfähige Bietfunktion beschrieben. Dabei wird nur das normale Spiel betrachtet, nicht jedoch das Nullspiel. Diesem widmet sich der spezielle Unterpunkt 4.5.4.

Evaluation gegen den eigenen Spieler

Nach den bisher eher theoretischen Betrachtungen stellt sich nun die Frage, wie stark die tatsächliche Spielstärke durch die vorgestellte Bietfunktion beeinflusst wird. Zu diesem Zweck habe ich zwei Spieler mit der lernfähigen Bietfunktion gegen zwei Gegner mit der statischen spielen lassen und dabei verschiedene Vereinfachungsstufen und Bietkorrekturen verwendet.

Alle bei den Testläufen verwendeten Hashtabellen wurden mit 100 Millionen Kartenblättern trainiert. Es wurden jeweils 5 Millionen Runden gespielt; ein Erfahrungswert aus den Tabellen wurde nur dann verwendet, wenn er aus mindestens 10 verschiedenen Einzelwerten gebildet wurde. Selbst bei der größten Vereinfachungsstufe (Ass bis Bube) wurden damit 98,80 % der erhaltenen Karten durch Erfahrungswerte abgedeckt.

Die genauen Ergebnisse bei unterschiedlichen Vereinfachungsstufen sind in Tabelle 4.4 aufgelistet. Neben der Bietkorrektur und den prozentual gewonnenen Spielen wird auch jeweils die durchschnittliche Abweichung zwischen Gebot und tatsächlich erhaltenen Stichen angegeben. Bei allen Vereinfachungsformen war stets der Durchlauf mit einer Bietkorrektur von -0,5 am erfolgreichsten.

Aufschlussreich ist die Veränderung der Siege durch die unterschiedlichen Vereinfachungen. Selbst die einfachste Form, bei der nur das Ass und die Anzahl der restlichen Karten gespeichert wird, schlägt die statische KI knapp. Erstaunlich ist hingegen das Ergebnis der nächsthöheren Stufe. Bereits wenn nur das Vorhandensein von Ass und König explizit gespeichert wird, schlägt der Spieler mit dieser Bietmethode die statische Funktion mit über 61 % Siegen. Eine solche Leistung hätte ich erst bei einer genaueren

Speicherung der Karten erwartet. Doch durch die Verwendung einer solchen, genauer gesagt *Ass bis Dame*, steigt die Leistung nur um ca. 1,5 Prozentpunkte weiter an. Das Maximum wird schließlich durch die gleiche Vereinfachung mit der zusätzlichen Speicherung des Pikbuben erreicht, was effektiv jedoch nur noch einer Steigerung um 1,16 % entspricht. Die Variante, bei der Ass bis Bube in allen Farben gespeichert wird, fällt hingegen wieder leicht ab und liegt damit zwischen den beiden letztgenannten. Aus diesem Grund habe ich auch keine noch detailliertere Vereinfachung mehr verwendet, da nicht mit einer Erhöhung der Spielleistung zu rechnen ist.

Nach dem Vergleich gegen die statische Bietfunktion bietet sich noch ein Testlauf der Vereinfachungsformen gegeneinandern an. Für diesen Vergleich wurden die selben Hashtabellen verwendet und ebenfalls 5 Millionen Runden gespielt. Als Bietkorrektur habe ich diesmal nur -0,5 benutzt, da dies für alle Varianten der beste Wert war. Die Ergebnisse sind in Tabelle 4.5 aufgelistet und entsprechen voll und ganz den Erwartungen. Die Reihenfolge der Spielleistungen hat sich gegenüber den vorigen Testläufen nicht verändert.

Entwicklung der Spielstärke

Für die eben gezeigten Resultate wurde jeweils eine mit vielen Erfahrungswerten aus den Trainingsrunden gefüllte Datenbasis verwendet. Aufschlussreich ist jedoch ebenso die Entwicklung der Spielleistung, wenn mit einem leeren Erfahrungsspeicher gegen einen Gegner gespielt und dabei trainiert wird. Zur Visualisierung habe ich die einzelnen Vereinfachungsformen jeweils 10 Millionen Runden gegen den statischen Spieler austragen lassen, jeweils mit einer Bietkorrektur von -0,5 und ohne einen Verwendungsgrenzwert.

Das Spiel ohne Grenzwert soll die Entwicklung der Spielstärke des jeweiligen Verfahrens ohne weitere Filterung verdeutlichen. Sobald der Schwellenwert auf (gut geeignete) 10 Vorkommnisse eines Kartenblatts gesetzt wird, beginnt die Spielleistung aller Methoden bei 50 % und fällt auch nie unter diesen Wert ab. Durch den Verzicht auf diese Maßnahme wird deutlich, wie die Ergebnisse bei der reinen Verwendung der gespeicherten Informationen aussehen, ohne verstärkt die statische Methode zu verwenden. Sobald die Siegesquote von 50 % erreicht ist existiert praktisch kein Unterschied mehr zur optimierten Spielweise mit einer Abfrage der Mindestvorkommnisse.

Die Steigerung der Spielleistung bezüglich der gespielten Runden zeigt Graph 4.3. Gut zu erkennen ist die schnelle Sättigung bei den groben Vereinfachungsformen. Bei einer genaueren Speicherung sind deutlich mehr Trainingsrunden erforderlich, was gut an der Stufe *Ass bis Bube* gesehen werden kann.

Da bei der dargestellten Entwicklung nur zwei Spieler mit einer lernfähigen Bietfunktion zum Einsatz kamen, sind am Ende des Durchlaufs insgesamt 20 Millionen Kartenblätter ausgewertet worden und damit deutlich weniger, als bei den Hashtabellen der bereits gezeigten Testreihen.

Die gleichen Daten sind in Graph 4.4 ver- bzw. entzerrt bezüglich der Abszisse, welche nun nicht mehr die insgesamt gespielten Runden darstellt, sondern die Einträge der jeweiligen Hashtabelle. Dadurch wird das Ergebnis etwas „linearisiert“. Dies soll vor allem zeigen, dass die Verfahren ungefähr linear mit den Einträgen der Hashtabellen, nicht jedoch der gespielten Runden, besser werden. Erst am Ende der Kurve beginnt diese ab-

zuflachen. Zudem kann auch hier gut erkannt werden, wieviele Einträge für ein adäquates Spielergebnis notwendig sind.

Evaluation gegen den GGZ-Spieler

Die bisherigen Ergebnisse zeigen eindeutig, dass die Bietfunktion auf der Basis von Erfahrungswerten die Spielleistung eines Spadesspielers deutlich steigern kann. Jedoch ist eine auf Testspielen basierende Evaluation gegen den selben Gegner mit einer statischen Bietmethode nicht sehr gut geeignet, eine allgemeine Aussage über die Spielstärke und den Gewinn durch die lernfähige Methode treffen zu können. In diesem Fall wird der Zugewinn an Stärke immer davon abhängen, wie gut die statische Bietfunktion arbeitet. Diese liefert zwar brauchbare Ergebnisse, wie an der Bietabweichung gesehen werden kann, jedoch ist der Vergleich mit einem gänzlich anderen Spieler aussagekräftiger.

Aus diesem Grund habe ich mich für eine Evaluierung gegen die KI des GGZ-Spadesservers entschieden. Da der Server mit den Clients über eine Netzwerkverbindung kommuniziert, ist die Performanz leider um mehrere Größenordnungen schlechter als ein interner Vergleich. Aus diesem Grund habe ich nicht alle Vereinfachungsformen, sondern nur die erfolgreichste (Ass bis Dame + Pikbube) mit verschiedenen Bietkorrekturen neben der statischen Bietfunktion getestet. Gespielt wurden jeweils 10.000 Runden, wobei der Server und die Clients auf dem selben Rechner liefen. Trotzdem dauerte der Durchlauf 3:45 Stunden, wobei alle Spiele ohne Zeitverlust parallel durchgeführt wurden. Zum Vergleich: Bei einem internen Testspiel werden pro Minute ca. eine Million Runden simuliert.

Da gegen einen anderen Gegner mit einer abweichenden Spielstärke nicht angenommen werden kann, dass der ideale Korrekturfaktor für die Gebote identisch zu einem eigenen Gegner ist, müssen auch hier wieder verschiedene Werte getestet werden. Als Vergleich ist auch das Ergebnis mit der statischen Bietfunktion angegeben. Tabelle 4.6 zeigt die Resultate des Testlaufs.

Zu Beachten ist natürlich, dass die Ergebnisse auf Grund der wenigen Spielrunden viel stärker vom tatsächlichen Mittelwert abweichen. Deutlich zu erkennen ist dies an der Bietabweichung zwischen den zwei Spielern eines Teams bzw. zwischen allen Ergebnissen des GGZ-Spielers. Trotzdem zeigen die Resultate, dass die Bietfunktion eindeutig die Spielstärke auch gegen fremde Gegner erhöht.

Bereits mit der statischen Bietmethode schlägt die eigens entwickelte KI den GGZ-Spieler knapp. Der Gewinn durch eine lernfähige Bietmethode ist im Verhältnis kleiner als bei den anderen Tests, eine Steigerung um 17 % mehr Spielgewinne ist jedoch durchaus ein positives Ergebnis.

4.5.3 Kartenblätter mit einer hohen Ansagedifferenz der unterschiedlichen Methoden

Interessant ist natürlich die Frage, bei welchen Kartenblättern die Differenz zwischen den Geboten der statischen und der eben vorgestellten Bietfunktion sehr groß ist.

Einen großen Unterschied gibt es vor allem bei Kartenklassen, die sehr selten vorkamen. In diesem Fall ist nur das Ergebnis eines oder sehr wenigen Spielen vermerkt, woraus sich

natürlich nicht auf die generelle Qualität der Karten schließen lässt (und auch von der Funktion nicht getan wird).

Beispielhaft habe ich die Einträge einer Hashtabelle mit der Vereinfachung *Ass bis Bube* nach der Differenz der gespeicherten Durchschnittswerte zu der statischen Vorhersage sortiert und die ersten 1.000 Einträge untersucht. Bei einer ausschließlichen Betrachtung von Einträgen mit einem Vorkommen von mindestens 10 reduziert sich die Menge bereits auf genau 18 Blätter, also gerade einmal 1,8 %. Der maximale Unterschied im untersuchten Bereich liegt dabei bei 3,4, der kleinste bei 3,0 (2,3 nach 10.000 Blättern).

Alle 18 Kartenblätter (und auch die meisten der nachfolgenden) weisen folgende Merkmale auf:

1. Eine hohe Anzahl von Trumpfkarten
2. Hohe und viele Farbkarten in einer Farbe
3. Keine Karten in zwei Farben oder keine Karten in einer und sehr wenige (meistens nur eine) Karten in der anderen Farbe

Als Beispiel dient der Eintrag mit der höchsten Differenz (3,4):

♠ *Dame* ♠ 6 ♠ 5 ♠ 4 ♠ 3 ♠ 2 | ♢ *Ass* ♢ *Dame* ♢ *Bube* ♢ 5 ♢ 4 ♢ 3 ♢ 2

Mit diesem Blatt wurden durchschnittlich 7,4 Stiche erreicht; die statische Bietmethode würde für die Karten 4 Stiche vorhersagen. Natürlich muss bei diesem Beispiel beachtet werden, dass es sich dabei nur um ein beispielhaft gewähltes Kartenblatt handelt, welches von der zugehörigen Äquivalenzklasse abgedeckt wird. Genauer gesagt entspricht dieses Beispiel allen Karten mit

- einer Dame und fünf zufällig verteilten Luschen in Pik sowie
- Ass, Dame, Bube und vier zufällig verteilten Luschen in einer beliebigen Farbe.

Mit dieser Kartenklasse wurde im Trainingslauf genau zehnmal gespielt (von 100 Millionen Kartenblättern insgesamt). Auch die anderen Karten mit einer hohen Ansagedifferenz kommen sehr selten vor, meistens lag die Anzahl der damit durchgeführten Spiele deutlich unter 100.

Bei den oft auftretenden Kartenklassen liegen die Unterschiede der Bietmethode wesentlich niedriger, oft sogar unterhalb von eins. In Tabelle 4.7 stehen die durchschnittlichen Abweichungen über alle gespeicherten Karten. Die Abweichung pro Äquivalenzklasse ignoriert die Häufigkeit ihres Auftretens, jede Klasse erhält das selbe Gewicht. Aussagekräftiger sind die Werte pro tatsächlich gespielter Kartenblätter, da hier sehr seltene Klassen (mit meist höheren Abweichungen) weniger Gewicht haben. Für beide Arten wurden die einzelnen Abweichungen aufaddiert und durch die entsprechende Gesamtzahl geteilt. Zusätzlich erfolgte diese Rechnung durch die betragsmäßige Addition. Diese Werte variieren nicht sehr stark, da die statische Bietmethode meistens weniger als die

möglichen Stiche ansagt. Dadurch bedingt sind die Werte alle positiv, da die statischen Gebote von den tatsächlich erzielten Stichen abgezogen werden. Eine visuelle Darstellung der Verteilung zeigt Graph 4.5. Es ist klar zu erkennen, dass die anfänglich sehr große Streuweite mit der Häufigkeit des Auftretens kontinuierlich abnimmt.

Zieht man von den Werten die (für ein erfolgreiches Spiel notwendige) Bietkorrektur der lernfähigen Methode ab (siehe 4.4.1) wird ersichtlich, dass die Abweichungen der tatsächlich angesagten Stiche beider Funktionen sehr klein ist. Da die Werte zusätzlich noch gerundet werden, wird sehr häufig von beiden Methoden der gleiche Wert angesagt.

Abweichungen bei der Ansage eines Nullspiels

Für eine Überprüfung der Nullspiele kann ähnlich vorgegangen werden. Die gespeicherten Erfahrungswerte habe ich nach der Gewinnhäufigkeit der Nullspiele sortiert und die erfolgreichsten Blätter mit der statischen Bietmethode bewertet. Die besten Nullblätter zeichnen sich durch folgende Eigenschaften aus:

1. Es sind keine oder nur sehr wenige Trumpfkarten vorhanden.
2. Eine Farbe ist frei oder nur mit sehr wenigen und niedrigen Karten besetzt.
3. In den besetzten Farben sind meistens mehr als drei Luschen vorhanden.

Blätter mit diesen Eigenschaften würden auch von der statischen Bietmethode mit Null bewertet. Der Vorteil der lernfähigen Methode beim Nullspiel ist die Möglichkeit, einen genauen Grenzwert festlegen zu können, bis zu welcher Gewinnchance gespielt wird. Legt man hier eine Wahrscheinlichkeit von 50 % und Mindestvorkommen von zehn zu Grunde, existieren in der herangezogenen Hashtabelle genau 14.806 Kartenblätter (von 666.255 erfassten Nullblättern insgesamt), mit denen ein Null gespielt werden würde.

Als Beispiel das Blatt, mit dem die meisten Nullspiele (55) gespielt, und auch alle gewonnen wurden:

♠ 2 | ♣ *Ass* ♣ *Dame* ♣ *Bube* ♣ 7 ♣ 6 ♣ 5 ♣ 4 ♣ 3 ♣ 2 | ♡ 4 ♡ 3 ♡ 2

Auch diese Karten sind natürlich nur ein Beispiel aus der korrespondierenden Äquivalenzklasse.

4.5.4 Ergebnisse des Nullspiels

Das Nullspiel wurde in den bisherigen Ergebnissen aus mehreren Gründen nicht berücksichtigt:

- Null ist ein sehr hochwertiges Sonderspiel. Ein Sieg des Nullspielers entspricht einem Fünftel der für einen Spielsieg erforderlichen Punkte (zusammen mit den Punkten des Partners meistens bereits mehr als einem Viertel).

- Die Vorhersage der Siegeswahrscheinlichkeit eines Nullspiels (für die eigene KI) ist, verglichen mit der normalen Bietfunktion, relativ unkompliziert.
- Die Wahrscheinlichkeit auf einen Sieg, ab der ein Null gespielt wird, beeinflusst maßgeblich die Anzahl der insgesamt gespielten und vor allem gewonnenen Spiele. Dabei zahlt sich eine hohe Risikobereitschaft bis zu einem gewissen Grad (ca. 50 % Siegeschance) aus.
- Das Team, das die meisten Nullspiele (bei ungefähr gleicher Siegeschance) spielt, gewinnt mit einer höheren Wahrscheinlichkeit das gesamte Spiel.
- Für das Nullspiel wird eine spezielle KI benötigt; eigentlich sogar eine eigene für den aktiven Nullspieler, dessen Partner und deren Gegner. Alleine durch die Nullansage eines Spielers verändern alle vier Spieler ihre Spielweise. Dabei sind die Grundziele dieser einzelnen Spielstrategien nicht mit einem normalen Spiel vergleichbar.
- Die Spielstärke der Null-KI (sowohl der eigenen als auch die der Gegenpartei) hat einen sehr großen Einfluss auf den Erfolg des Nullspiels. Mit einer starken (eigenen) KI können demnach mehr Nullspiele, mit einem starken Gegner hingegen weniger Nullspiele angesagt werden.

Im Endeffekt entscheidet über ein erfolgreiches Nullspiel zum größten Teil die Stärke der Spiellogik, nur zu einem relativ kleinen Teil jedoch die verwendete Bietmethode. Zur Leistungsmessung der lernfähigen Bietmethoden wurden bisher reale Spielergebnisse, genauer gesagt die Anzahl der gewonnenen Spiele, herangezogen. Dies ist ein guter Vergleich für die eigentliche Bietfunktion, da alle Spieler eines Teams für sämtliche Spiele die selbe KI und damit auch Spielstrategie benutzen. Verändert wurde in den Testreihen nur die jeweils verwendete Bietfunktion, weshalb die Ergebnisse auch explizit deren Leistung widerspiegeln. Die Aussage der Versuche ist je nach der Art des durchgeführten Vergleichs die

1. Leistung der Bietfunktion bezüglich einer KI,
   identische KI für alle Spieler, verschiedene Bietfunktionen je Team
2. Leistung der KI bezüglich einer Bietfunktion,
   identische Bietfunktion für alle Spieler, verschiedene KIs je Team
3. Leistung der Kombination aus Spiellogik und Bietfunktion.
   identische Kombinationen aus Bietfunktion und KI je Team

Die meisten durchgeführten Versuche entsprechen Punkt eins. Die dabei entstandenen Ergebnisse zeigen den *absoluten Einfluss* auf die Spielleistung durch verschiedene Bietfunktionen bei ansonsten identischen Spielern. Die Testreihen gegen den GGZ-Spieler (Punkt drei) zeigen hingegen den *relativen Einfluss* der Bietfunktion auf die Spielleistung, da der GGZ-Spieler neben einer anderen Bietfunktion auch eine andere KI verwendet.

Bei einer Hinzunahme des Nullspiels zu den Testspielen kämen zwei weitere Faktoren neben der KI und der Bietfunktion hinzu, nämlich

1. die Ansagefunktion des Nullspiels sowie

2. die Spielstärke der verschiedenen Null-KIs (Nullspieler, Partner und Gegner).

Wie bereits erwähnt ist die Wertigkeit des Nullspiels im Vergleich zu normalen Spielen sehr hoch, so dass dieses sehr oft spielentscheidend ist. Während eine Funktion zur Ansage von Null auf Grund der binären Entscheidung wesentlich einfacher als die normale Bietfunktion aufgebaut sein kann, trifft für die Null-KIs, auch durch die nun verteilten Rollen der Mitspieler, prinzipiell das Gegenteil zu.

Durch die Berücksichtigung des Nullspiels würden viele Spiele von der Partei mit der besseren Null-KI gewonnen, auch wenn diese eine schlechtere normale Bietfunktion benutzt. Dadurch verlören die Ergebnisse ihre komplette Aussagekraft bezüglich der normalen Bietfunktion, ohne dabei jedoch die Leistung der Nullbietfunktion zu repräsentieren. Mit normalen Spielergebnissen könnte nur noch die Gesamtheit des Spielers bewertet werden, nicht mehr jedoch einzelne Komponenten.

Zur Verdeutlichung des Stellenwerts des Nullspiels zeigt Tabelle 4.8 drei Spielergebnisse der statischen Bietfunktion gegen die in diesem Kapitel vorgestellte Bietmethode mit der Vereinfachung *Ass bis Dame + Pikbube*. Die Hashtabelle enthielt 100 Millionen ausgewertete Kartenblätter für das Null- als auch für das normale Spiel. Für den Vergleich wurden 5 Millionen Runden gespielt.

Das Ergebnis aus der ersten Zeile entspricht einem Spiel komplett ohne Null. Für die Ergebnisse der zweiten Zeile konnten beide Bietfunktionen ein Null ansagen, für den letzten Durchlauf durfte nur noch die statische Bietfunktion ein Null spielen (wobei die andere Partei weiterhin als vollwertiger Nullgegner agierte).

Die Ergebnisse verdeutlichen folgende Besonderheiten des Nullspiels:

1. Der Leistungsunterschied zwischen den zwei Bietfunktionen verringert sich deutlich, sobald auch Null gespielt wird.

2. Verzichtet ein Team komplett auf das Nullspiel, so sinkt die Gewinnaussicht eklatant. In diesem Beispiel sogar deutlich unter 50 %.

Da der Spieler selbst entscheidet, bis zu welchem Risiko er ein Null ansagt oder auf ein normales Spiel ausweicht, sagt die Siegesquote aller durchgeführten Nullspiele alleine nichts über die Güte der Bietfunktion aus. Es muss deshalb ein Kriterium zur Bewertung der Nullbietmethode gefunden werden, mit dem diese objektiv bewertet werden kann.

Da es von Vorteil ist, möglichst viele (erfolgreiche) Nullspiele zu spielen, kann die Anzahl der angesagten Nullspiele bei einer (ungefähr) gleich hohen Siegesquote als Qualitätsmerkmal dienen. Für diesen Vergleich wurden für alle Vereinfachungsstufen 5 Millionen Runden gegen zwei Gegner mit statischer Bietmethode durchgeführt. Der Grenzwert für die Ansage eines Nullspiels lag bei 50 % Siegeswahrscheinlichkeit. Tabelle 4.9 zeigt die ermittelten Ergebnisse. Von der ersten Vereinfachung abgesehen ist die Siegesquote relativ konstant, so dass ein Vergleich der absoluten Nullspiele möglich ist. Zusätzlich sind in der Tabelle die insgesamt gewonnenen Spiele eingetragen, wobei der Einfluss der Nullansagefunktion nicht von der normalen Bietfunktion unterschieden werden kann.

Sehr deutlich ist zu erkennen, dass mit dem steigenden Detailgrad der Vereinfachung die Anzahl der Nullspiele zunimmt. Der Grund für diese Veränderung des Ansageverhaltens ist leicht zu erkennen:

Die verwendete Vereinfachung wurde für die normale Bietfunktion und nicht zur Ansage eines Nullspiels entwickelt. Während für die Einschätzung der möglichen Stiche vor allem die hohen Karten des Blatts interessant sind (weshalb auch nur diese explizit gespeichert werden), sind die Anforderungen des Nullspiels an die Karten praktisch komplementär. Sobald jedoch hohe nicht mehr von niedrigen Karten unterschieden werden, wird eine Nullansage unsicher. In der Praxis werden somit (aus der Sicht des Nullspiels) vollkommen unterschiedliche Kartenblätter zu einer Äquivalenzklasse zusammengefasst. Durch die Zusammenfassung von erfolgreichen mit nicht erfolgreichen Kartenblättern zu einer Klasse ist die letztendlich gespeicherte Wahrscheinlichkeit auf einen Nullsieg für diese relativ gering. Aus diesem Grund werden bei gleichem Grenzwert mit der Vereinfachungsstufe *Ass* weniger als 60 % der Nullspiele der Stufe *Ass bis Bube* angesagt.

Trotz dieser Unzulänglichkeit der benutzten Methode für das Nullspiel ist zumindest bei hohem Vereinfachungsgrad dennoch eine gute Vorhersage möglich. Wird z.B. Ass bis Bube separat gespeichert, bleiben nur relativ ungefährliche niedrige Karten übrig. Liegen von diesen sogar mehrere vor, ist ein Nullspiel meistens relativ gefahrlos möglich.

Ideal wäre zweifellos eine separate Vereinfachungsfunktion der Handkarten für den Zweck der Nullansage. Ob sich dieser Aufwand in der Praxis jedoch auszahlen würde kann nicht leicht beantwortet werden. Einen ersten Hinweis liefern die Ergebnisse des Nullspiels mit einem künstlichen neuronalen Netz (siehe 5.5.3).

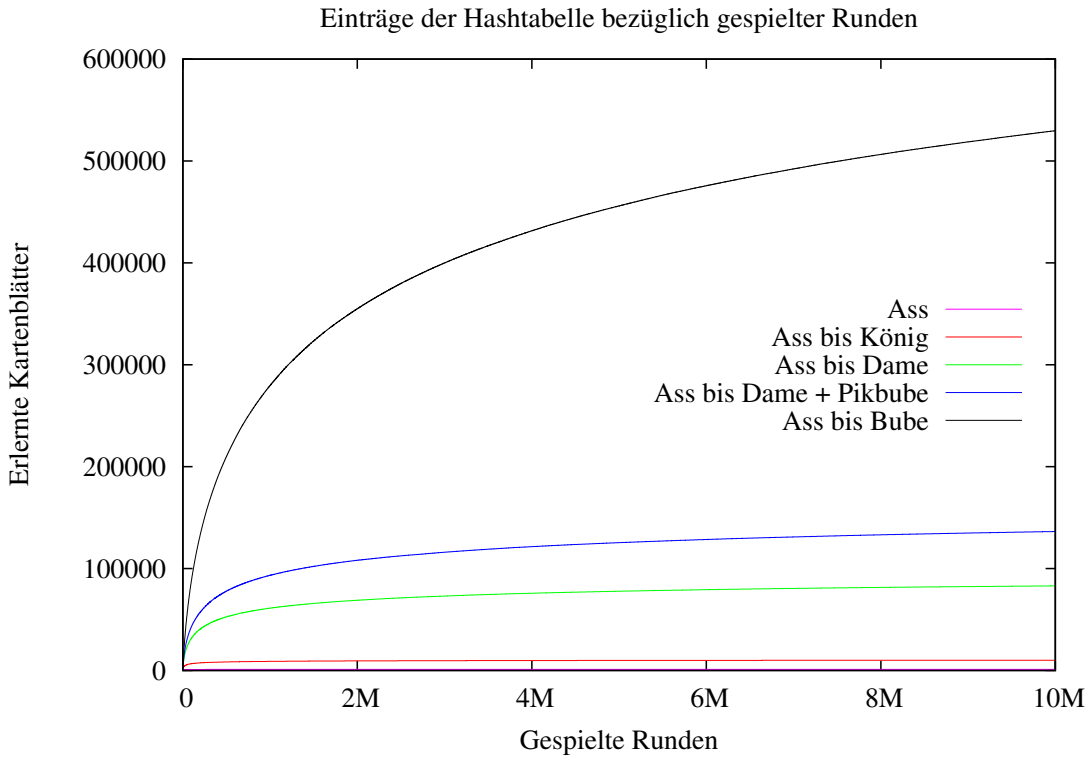

Abbildung 4.1: Anzahl der bisher aufgetretenen unterschiedlichen Kartenklassen in Bezug zu den gespielten Runden

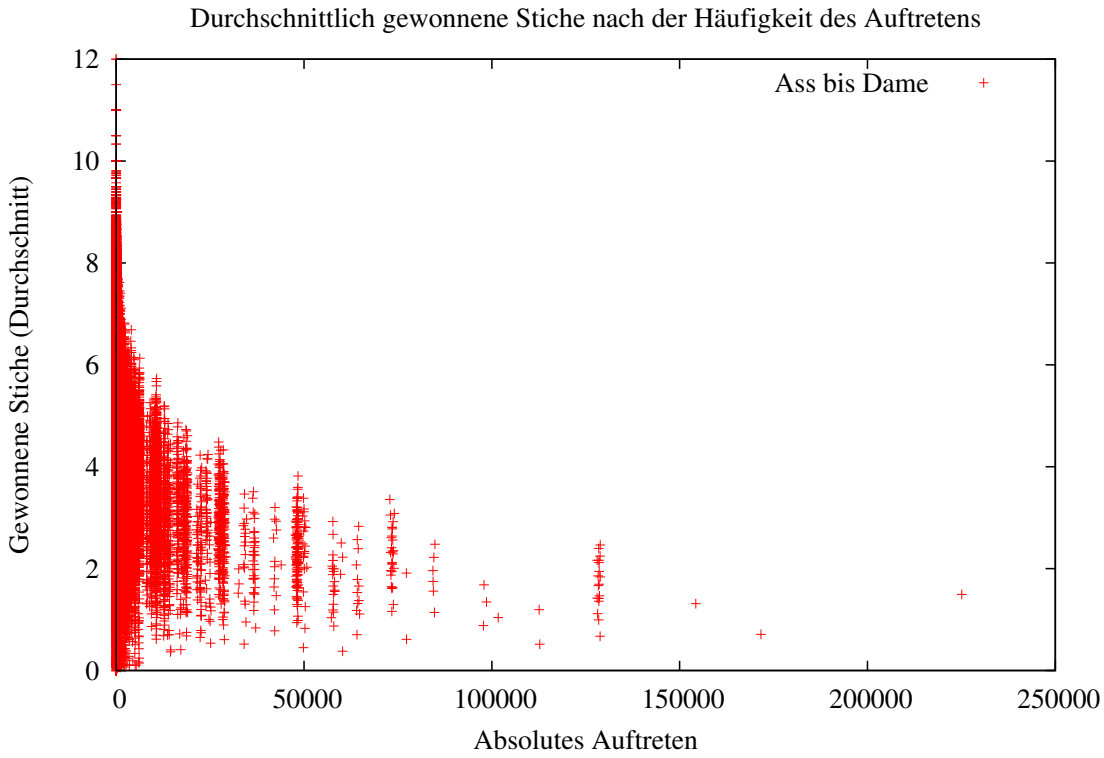

Abbildung 4.2: Durchschnittlich gewonnene Stiche gegenüber dem absoluten Auftreten dieser Kartenvariation. Vereinfachungstyp: Ass bis Dame

Bietkorrektur	Siege	Abweichung (Betrag)	Stat. Abw. (Betrag)
-0,3	46,21 %	0,31 (0,88)	0,54 (0,94)
-0,4	50,32 %	0,40 (0,91)	0,54 (0,94)
-0,5	52,07 %	0,48 (0,93)	0,54 (0,94)
-0,6	51,75 %	0,60 (0,98)	0,54 (0,94)
-0,7	47,81 %	0,72 (1,03)	0,54 (0,94)

(a) Vereinfachung: Ass

Bietkorrektur	Siege	Abweichung (Betrag)	Stat. Abw. (Betrag)
-0,3	56,91 %	0,31 (0,78)	0,54 (0,94)
-0,4	60,30 %	0,38 (0,81)	0,54 (0,94)
-0,5	61,38 %	0,47 (0,84)	0,54 (0,94)
-0,6	60,46 %	0,57 (0,89)	0,54 (0,94)
-0,7	57,00 %	0,67 (0,94)	0,54 (0,94)

(b) Vereinfachung: Ass und König

Bietkorrektur	Siege	Abweichung (Betrag)	Stat. Abw. (Betrag)
-0,3	57,96 %	0,29 (0,76)	0,54 (0,94)
-0,4	61,68 %	0,38 (0,79)	0,54 (0,94)
-0,5	62,84 %	0,47 (0,83)	0,54 (0,94)
-0,6	61,65 %	0,57 (0,87)	0,54 (0,94)
-0,7	58,32 %	0,66 (0,92)	0,54 (0,94)

(c) Vereinfachung: Ass bis Dame

Bietkorrektur	Siege	Abweichung (Betrag)	Stat. Abw. (Betrag)
-0,3	59,08 %	0,29 (0,75)	0,54 (0,94)
-0,4	62,71 %	0,38 (0,78)	0,54 (0,94)
-0,5	63,57 %	0,48 (0,82)	0,54 (0,94)
-0,6	62,60 %	0,57 (0,86)	0,54 (0,94)
-0,7	59,28 %	0,66 (0,91)	0,54 (0,94)

(d) Vereinfachung: Ass bis Dame + Pikbube

Bietkorrektur	Siege	Abweichung (Betrag)	Stat. Abw. (Betrag)
-0,3	58,45 %	0,28 (0,75)	0,54 (0,94)
-0,4	61,99 %	0,38 (0,78)	0,54 (0,94)
-0,5	63,20 %	0,47 (0,82)	0,54 (0,94)
-0,6	61,98 %	0,56 (0,86)	0,54 (0,94)
-0,7	58,65 %	0,65 (0,91)	0,54 (0,94)

(e) Vereinfachung: Ass bis Bube

Tabelle 4.4: Spielergebnisse gegen die statische Bietfunktion

Team 1	Siege	Abweichung	Team 2	Siege	Abweichung
Ass	40,97 %	0,48 (0,93)	Ass - König	59,03 %	0,47 (0,84)
Ass	39,45 %	0,48 (0,93)	Ass - Dame	60,55 %	0,47 (0,83)
Ass	38,87 %	0,48 (0,93)	A. - D. + ♠B	61,13 %	0,48 (0,82)
Ass	39,49 %	0,48 (0,93)	Ass - Bube	60,51 %	0,47 (0,82)
Ass - König	48,51 %	0,47 (0,84)	Ass - Dame	51,49 %	0,47 (0,83)
Ass - König	47,83 %	0,47 (0,84)	A.- D. + ♠B	52,17 %	0,48 (0,82)
Ass - König	48,37 %	0,47 (0,84)	A. - Bube	51,63 %	0,47 (0,82)
Ass - Dame	49,44 %	0,47 (0,83)	A. - D. + ♠B	50,56 %	0,48 (0,82)
Ass - Dame	49,92 %	0,47 (0,83)	Ass - Bube	50,08 %	0,47 (0,82)
A. - D. + ♠B	50,70 %	0,48 (0,82)	Ass - Bube	49,30 %	0,47 (0,82)

Tabelle 4.5: Spielergebnisse der verschiedenen Vereinfachungsformen im direkten Vergleich

Bietf.	Bietk.	Siege	Abweichung S1/S3	Abw. S2/S4 (GGZ)
Stat.	-	53,54 %	0,59 (1,01) / 0,62 (1,17)	0,12 (1,26) / 0,06 (1,24)
Hasht.	-0,3	61,24 %	0,27 (0,77) / 0,36 (1,00)	0,19 (1,30) / 0,14 (1,24)
Hasht.	-0,4	62,40 %	0,40 (0,83) / 0,45 (1,05)	0,18 (1,26) / 0,13 (1,23)
Hasht.	-0,5	62,50 %	0,51 (0,89) / 0,57 (1,07)	0,12 (1,27) / 0,09 (1,23)
Hasht.	-0,6	62,67 %	0,63 (0,94) / 0,68 (1,11)	0,10 (1,24) / 0,07 (1,18)
Hasht.	-0,7	60,24 %	0,74 (1,01) / 0,80 (1,19)	0,06 (1,23) / 0,03 (1,20)

Tabelle 4.6: Spielergebnisse gegen den GGZ-Spieler

Berechnungsgrundlage	Abweichung
Äquivalenzklasse	0,335
Äquivalenzklasse (Betrag)	0,747
gespielter Blätter	0,537
gespielter Blätter (Betrag)	0,671

Tabelle 4.7: Durchschnittliche Abweichungen der stat. Bietfunktion über alle gespeicherten Kartenblätter (Vereinfachung: Ass bis Bube, Kartenblätter: 100 Millionen)

Team 1	Siege	Nullspiele (Siege)	Team 2	Siege	Nullspiele (Siege)
Hasht.	63,57 %	-	Stat.	36,43 %	-
Hasht.	59,45 %	936.004 (67,81 %)	Stat.	40,55 %	1.171.470 (61,49 %)
Hasht.	46,44 %	-	Stat.	53,56 %	1.170.349 (59,36 %)

Tabelle 4.8: Spielergebnisse mit und ohne Nullspiel

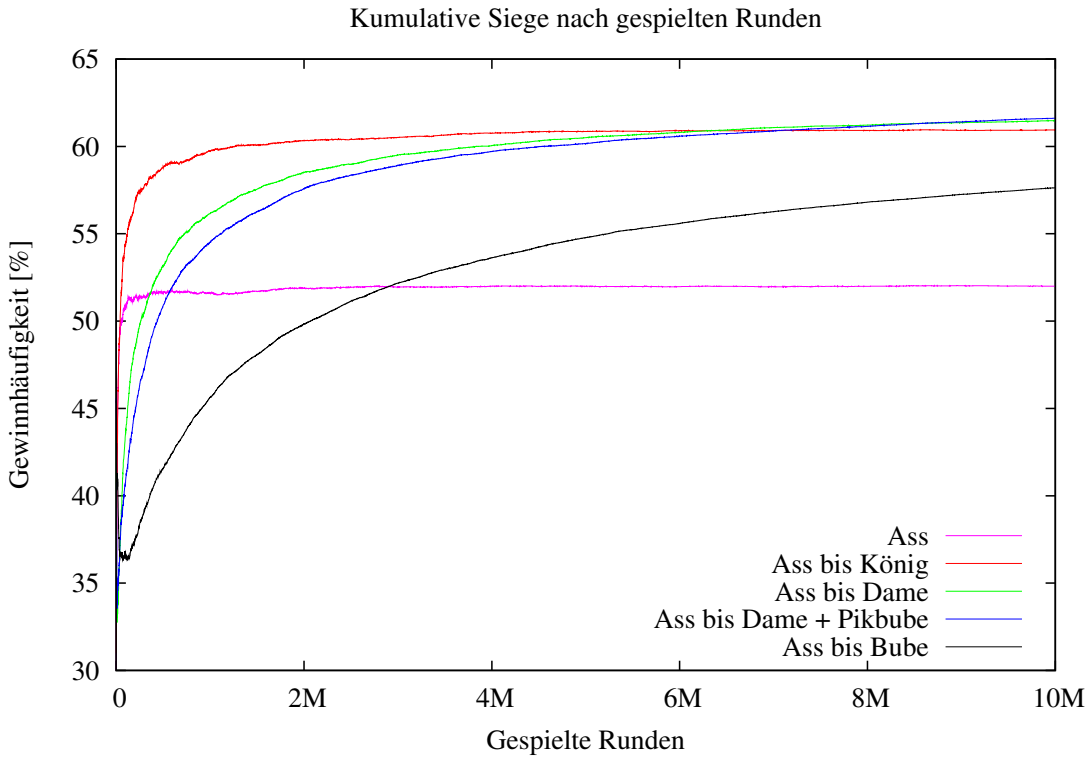

Abbildung 4.3: Gewinnhäufigkeit der lernfähigen KI nach der Anzahl der gespielten Runden

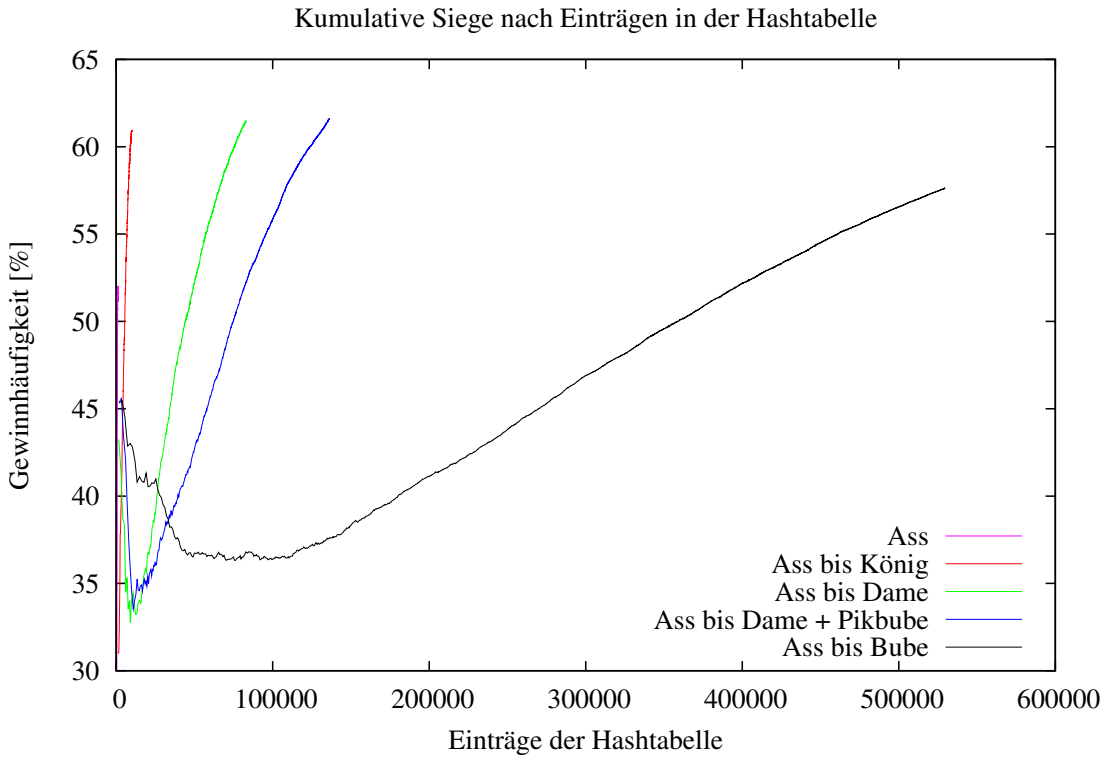

Abbildung 4.4: Gewinnhäufigkeit der lernfähigen KI nach der Anzahl der bisher erlernten unterschiedlichen Kartenblätter

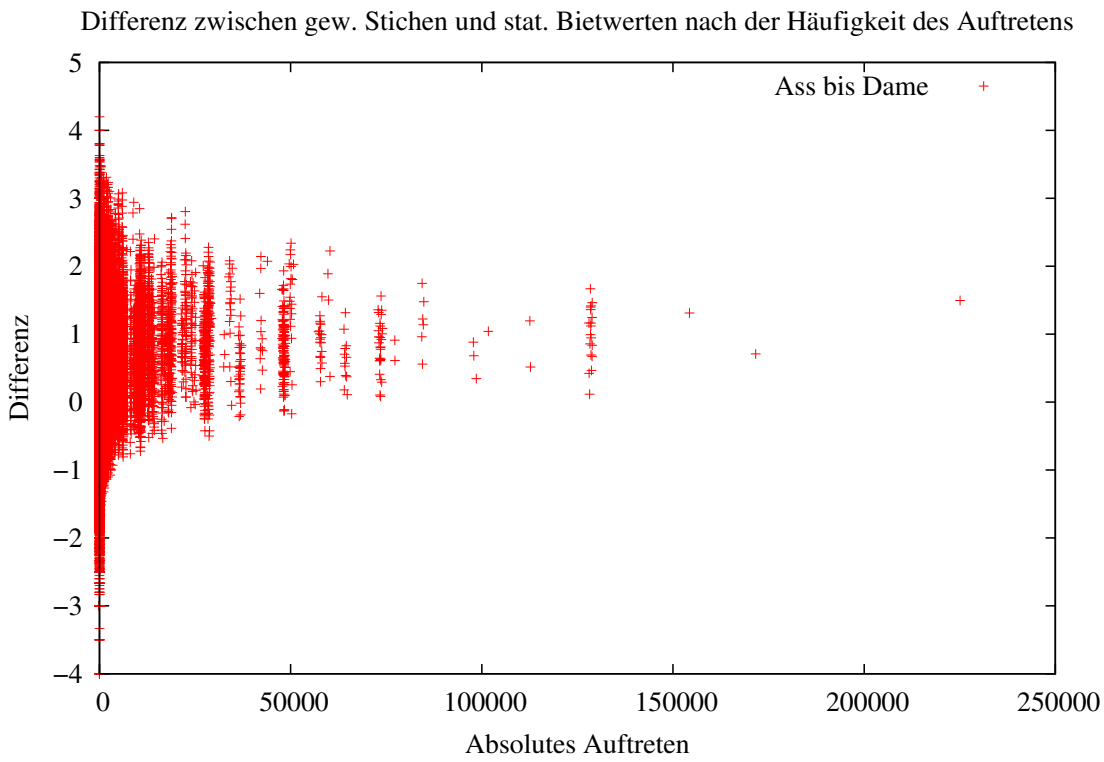

Abbildung 4.5: Differenz zwischen den tatsächlich gewonnenen Stichen (+) und der Ansage der statischen Bietmethode (-) gegenüber dem absoluten Auftreten der Kartenvariation. Vereinfachungstyp: Ass bis Dame

Team 1	**Siege**	**Nullspiele (Siege)**	**Team 2**	**Siege**	**Nullspiele (Siege)**
Ass	47,46 %	571.142 (63,68 %)	Stat.	52,54 %	1.168.765 (60,59 %)
Ass - König	54,07 %	610.709 (67,25 %)	Stat.	45,93 %	1.170.190 (60,74 %)
Ass - Dame	57,94 %	859.474 (67,25 %)	Stat.	42,06 %	1.170.236 (61,37 %)
A. - D. + ♠B	59,26 %	934.202 (67,79 %)	Stat.	40,74 %	1.169.846 (61,50 %)
Ass - Bube	59,96 %	963.220 (68,60 %)	Stat.	40,04 %	1.168.731 (61,48 %)

Tabelle 4.9: Spielergebnisse mit und ohne Nullspiel

5 Bieten durch neuronale Netze

Das eben vorgestellte Verfahren verbessert seine Bietleistung durch das simple Abspeichern und Auswerten von Erfahrungswerten zu bestimmten Karten(-klassen), es handelt sich jedoch nicht um ein echtes Lernverfahren. Obwohl die Ergebnisse vielversprechend sind, soll an dieser Stelle ein etabliertes Mittel aus dem Bereich des maschinellen Lernens zur Lösung des Bietproblems vorgestellt werden: Künstliche neuronale Netze (KNN).

Die Nachteile des bisherigen Verfahrens liegen auf der Hand:

- Es können keine bisher unbekannten Kartenblätter bewertet werden.
- Es sind sehr viele Trainingsdaten für eine ausreichende Erfassung der Kartenklassen erforderlich.

Aus diesem Grund wird im folgenden Abschnitt die Verwendung von künstlichen neuronalen Netzen für eine hochwertige Bietmethode erläutert. Zu Beginn wir das generelle Konzept neuronaler Netze kurz vorgestellt und anschließend die konkrete Verwendung im Rahmen dieser Arbeit aufgezeigt. Abgeschlossen wird das Kapitel durch eine Gegenüberstellung der erzielten Ergebnisse mit den Spielleistungen der vorherigen Methoden.

5.1 Kurzvorstellung

Künstliche neuronale Netze basieren auf der Vernetzung künstlicher Neuronen, oftmals von den bereits 1943 vorgestellten McCulloch-Pitts-Neuronen oder Abwandlungen von diesen. Es sind jedoch auch einige andere Modelle beschrieben und werden in der Praxis verwendet. KNNs sind seit Jahren Forschungsobjekt und deshalb auch ausführlich in der Literatur beschrieben. Ein erster Überblick inkl. weiterführenden Literaturverweisen findet sich unter [17].

Prinzipiell ist ein KNN eine stark vereinfachte Nachbildung eines biologischen neuronalen Netzes, also die Verbindung der Nervenzellen im Gehirn und Rückenmark, jedoch steht im Vordergrund eine Abstraktion der Art und Weise der biologischen Informationsverarbeitung und keine wirklichkeitsgetreue Simulation.

Der genaue Aufbau eines Netzes, vor allem die Anzahl der Eingangs- und Ausgangsknoten sowie die Anzahl der Neuronen in der oder den mittleren Schichten, muss der jeweiligen Aufgabe angepasst sein. Nach der erstmaligen Konstruktion des Netzes wird dieses mit realen Daten der zu lösenden Problemstellung trainiert. In dieser Phase „lernt“ das KNN, wobei folgende Veränderungen (theoretisch) vorgenommen werden können:

- Aufbau von neuen Verbindungen zwischen einzelnen Neuronen.
- Löschen von bestehenden Verbindungen.
- Anpassung der Gewichtung der Verbindung zwischen zwei Neuronen. Die Gewichtung bestimmt dabei die Stärke des Einflusses, den die Eingaben des Vorgängerneurons auf die Ausgabe des aktuellen Neurons haben. Ist die Gewichtung negativ, so wirkt eine Eingabe hemmend (inhibitorisch), ansonsten erregend (exhibitorisch).
- Anpassung des Schwellenwerts der Neuronen. Dieser Wert entspricht einem Grenzwert der Eingaben, der für eine Aktivierung des Neurons erreicht oder überschritten werden muss.
- Hinzufügen oder Entfernen von Neuronen.

In der Praxis wird in der Trainingsphase vor allem die Verbindungsgewichtung und die Schwellenwerte der Neuronen angepasst. Wird die Gewichtung auf null gesetzt, so existiert praktisch keine Verbindung mehr zwischen zwei Neuronen, weshalb diese Anpassung implizit vorgenommen werden kann.

Für das eigentliche Lernverfahren gibt es drei verschiedene Möglichkeiten, welche wiederum viele unterschiedliche Lernvarianten enthalten. Normalerweise kann jedes Netz in eine der drei Klassen eingeteilt werden:

Überwachtes Lernen (Supervised learning) Bei diesem Verfahren müssen Trainingsdaten mit bekannten Ergebnissen vorliegen. Das Netz erhält als Eingabe ein Testmuster und produziert daraufhin eine Ausgabe. Diese wird mit dem korrekten Ergebnis verglichen und der Fehler berechnet. Auf dieser Basis können schließlich Veränderungen am Netz vorgenommen werden.

Unüberwachtes Lernen (Unsupervised learning) wird häufig zur Komprimierung und Klassifikation bzw. Clusteranalyse eingesetzt. Das Netz erhält Eingaben ohne bekannte Ausgabe und verteilt diese auf verschiedene (nicht vorgegebenen) Klassen.

Bestärkendes Lernen (Reinforcement learning) bezeichnet eine Lernform, bei der üblicherweise keine expliziten Trainingsdaten zur Verfügung stehen. Stattdessen agiert ein sog. Agent direkt mit seiner Umgebung (z.B. einer Spielsituation) und trifft Entscheidungen, die den Zustand seiner Umgebung verändern und für welche er eine Belohnung erhält. Das Ziel des Agenten ist die Maximierung des zu erwarteten Gewinns über eine gewisse Zeitspanne (z.B. die Dauer eines Spiels).

Künstliche neuronale Netze sind sehr gut für Problemstellungen geeignet, für die kein oder nur geringes Expertenwissen vorliegt, lediglich die Topologie des Netzes und die Kodierung der Eingangsdaten muss der Problemstellung angepasst sein. Aus diesem Grund ist der Einsatz eines KNNs zur Klassifikation von Spadeskarten vielversprechend.

5.1.1 Überanpassung (Overfitting)

Für das Training eines KNNs werden sämtliche Trainingsdaten nacheinander an die Eingangsknoten angelegt und je nach verwendeter Lernmethode auf der Basis der Ausgabe Veränderungen vorgenommen. Normalerweise wird dieser Vorgang in mehreren Schritten, Epochen genannt, wiederholt, evtl. mit einer zunehmenden Abschwächung des Lerneffekts. Idealerweise wird die Qualität der Ausgaben nach jedem Zyklus an unabhängigen Testdaten überprüft. Die Erkennungsleistung des Netzes auf den Testdaten steigt mit der Anzahl der Epochen bis zu einer Sättigungsphase zunächst an, fällt jedoch bei einer Fortsetzung des Trainings oftmals wieder ab.

In diesem Fall wurden Merkmale der Trainingsdaten erlernt, die jedoch nicht repräsentativ für das Problem als ganzes sind. Bildlich gesprochen wurden die Trainingsdaten auswendig gelernt, anstatt ausgehend von diesen das eigentliche Problem zu abstrahieren.

Um dies zu vermeiden muss der Lernvorgang unter Zuhilfenahme von Test- oder auch zusätzlichen Validierungsdaten kontrolliert werden. Das Training wird im Allgemeinen dann abgebrochen, wenn der Fehler aus diesen Daten wieder zu steigen beginnt.

5.2 Verwendete Lernmethode

Als erstes muss eine der Problemstellung entsprechende Lernmethode ausgewählt werden. Für Spiele wird häufig bestärkendes Lernen angewendet und das Netz durch mehrfaches Spielen trainiert. Für diese Arbeit soll hingegen nicht das komplexe Spiel als solches, sondern nur die Bietphase erlernt werden. Dies ist ein reines Klassifikationsproblem: Alle möglichen Handkarten müssen exakt einer der 14 möglichen Klassen zugeordnet werden. Die Klassen entsprechen in diesem Fall den angesagten Stichen von 0 bis 13. Über das Nullspiel wird aus den gleichen Gründen wie schon in 4.4.2 dargelegt separat entschieden (siehe 5.4.2). Zudem können beliebige Mengen von Trainingsdaten erzeugt werden, sofern man von einem direkten Training gegen menschliche Spieler absieht.

Da alle notwendigen Voraussetzungen erfüllt sind, bietet sich als Lernverfahren überwachtes Lernen an, genauer gesagt ein Netz mit Fehlerrückführung (Backpropagation). Bei diesem Verfahren wird ein Eingabemuster an das KNN angelegt und durch das Netz propagiert. Die berechnete Ausgabe wird mit der erwarteten verglichen und die Differenz als Fehler des Netzes gewertet. Dieser Fehler wird schließlich rückwärts, von der Ausgabe- zur Eingabeschicht, durch das Netz propagiert, wobei die Gewichtungen der Neuronenverbindungen abhängig von ihrem jeweiligen Einfluss auf den entstandenen Fehler angepasst werden. Durch diese Maßnahme wird die Ausgabe bei einem erneuten Anlegen der Eingabedaten der korrekten Ausgabe angenähert.

5.3 Konstruktion und Training

Um ein künstliches neuronales Netz in einer eigenen Applikation zu verwenden, muss dieses nicht von Grund auf neu implementiert werden. Mit Weka [24] existiert ein quelloffenes Framework zur Integration von verschiedenen Algorithmen des maschinellen Ler-

nens und des Data-Minings in beliebige Anwendungen. Neben der einfachen Integration zeichnet sich Weka durch eine Reihe von mitgelieferten Werkzeugen zur Konstruktion, Überprüfung und zum Training der verschiedenen Klassifikatoren aus.

In diesem Abschnitt möchte ich die Vorgehensweise zur Erstellung und zum Training eines KNNs mit dem *Weka Explorer* beschreiben.

5.3.1 Format der Trainingsdaten

Anforderungen und Verhalten von Weka

Die Trainingsdaten müssen im *Attribute-Relation File Format* (ARFF)[1] vorliegen, um von Weka verarbeitet zu werden. Dabei handelt sich um eine ASCII-Textdatei mit einer Liste von Attributen und zugehörigen Datensätzen, Instanzen genannt. Der Datentyp der Attribute ist entweder numerisch, nominal, ein String oder vom Typ Datum. Für die hier benötigte Klassifikation sind nur die ersten beiden Datentypen interessant, String und Datum werden nicht benötigt.

Numerische Werte sind Gleitkommazahlen und werden gewöhnlich für die Eingabedaten verwendet. Es können auch Ganzzahlen angegeben werden, diese werden jedoch intern umgewandelt. Ist der Ausgabewert numerisch, so gibt das KNN direkt einen rationalen Wert aus. Ist als Ausgabe eine Ganzzahl gefordert, muss der Rückgabewert gerundet oder andersweitig umgewandelt werden.

Nominale Werte sind selbst definierte Klassen, wobei letztendlich nur die Namen festgelegt werden. Ist der Ausgabewert nominal definiert, so gibt das KNN für jede mögliche Klasse die Größe der Übereinstimmung zwischen null und eins aus, wobei die Summe immer eins ergibt. Ist als Ausgabe eine bestimmte Klasse erforderlich, wird diese folglich durch den Maximalwert bestimmt.

Generieren von Datensätzen

Die Erzeugung von Trainingsdaten ähnelt sehr der bereits im vorigen Kapitel beschriebenen Vorgehensweise. Obwohl einer der Vorteile eines KNNs darin besteht, im Verlauf der Konditionierung automatisch die relevanten Daten zu erkennen und unwichtige durch eine sehr geringe Gewichtung praktisch zu eliminieren, muss auch hier auf eine vernünftige Darstellung der Karten geachtet werden. Die Eingangsdaten müssen dem konkreten Problem angepasst und nach Möglichkeit redundanzfrei kodiert werden.

Prinzipiell eignet sich deshalb die unter 4.1.1 beschriebene Vereinfachung ebenfalls für diesen Zweck. Um der Charakteristik des KNNs gerecht zu werden, wird die höchste Detailstufe, also die exakte Speicherung von Ass bis Bube, verwendet. Zusätzlich wird eine Vereinfachungsform getestet, die zwar die drei Farben Kreuz, Herz und Karo sortiert, dabei jedoch alle Karten explizit speichert. Dadurch werden die Karten ohne Informationsverlust an das Netz übergeben. Ein direkte Verwendung des Kartenblatts ohne Sortierung erscheint hingegen wenig sinnvoll, da durch dieses Vorgehen zwar der Zustandsraum enorm vergrößert wird, nicht jedoch der Informationsgehalt der Daten.

Den genauen Aufbau eines Datensatzes zeigt Tabelle 5.1.

[1] http://www.cs.waikato.ac.nz/~ml/weka/arff.html

Nr.	1	2	3	4	5	...	21
Name	Ass	König	Dame	Bube	Anzahl	...	Stiche
Werte	0 - 1	0 - 1	0 - 1	0 - 1	0 - 9	...	0 - 13

Tabelle 5.1: Übergabeformat der vereinfachten Karten an Weka. Pik wird in 1-5 gespeichert, die drei anderen Farben sortiert in 6-10, 11-15 und 16-20

5.3.2 Numerische oder nominale Klassifizierung?

Alle bei Spades direkt vorkommenden Werte sind ganzzahlig, dies resultiert bereits aus der generellen Charakteristik eines Kartenspiels mit atomaren Spielkarten. Aus diesem Grund ist es unmöglich, eine nicht ganzzahlige Anzahl von Stichen zu erhalten oder ebensolche Gebote zu tätigen. Auf den ersten Blick erscheint deshalb die Verwendung des Nominalattributs für die Zielklasse ideal zu sein. In der Praxis werden damit jedoch keine guten Spielergebnisse erreicht. Die Ursache ist, wie bei der im vorigen Kapitel erläuterten Methode, in der Bepunktung des Spiels zu suchen. Das Verhalten des KNNs bei einer nominalen Einordnung liefert ähnliche Resultate wie das echte Runden bei dem Spiel mit reinen Erfahrungswerten, und liegt damit deutlich unterhalb der Spielleistung der statischen Methode. Mit der numerischen Klassifikation und der bereits in 4.4.1 beschriebenen Korrektur werden dagegen wesentlich bessere Leistungen ermöglicht.

5.3.3 Konkrete Erzeugung des KNNs

Zur Erstellung eines neuen KNNs mit dem Weka Explorer (WE) müssen etliche Entscheidungen bezüglich der Topologie und des Trainings getroffen werden. Den eigentlichen Aufbau des Netzes kann der WE auf Wunsch automatisch übernehmen (was auch genutzt wurde). Die folgende Liste zeigt die wichtigsten Parameter:

- Kodierung und Umfang der Trainings- und Testdaten
- Anzahl der versteckten Schichten (Hidden Layers)
- Anzahl der Trainingszyklen (Epochen)
- Bestimmung der Lernrate (der Umfang, mit dem die Verbindungsgewichtungen angepasst werden) und des Momentums
- Dämpfung der Lernrate nach jeder Epoche
- Bestimmung der Testweise des trainierten KNNs

Manche dieser Einstellungen haben einen großen Einfluss auf die Leistung des resultierenden KNNs, während andere das Ergebnis kaum oder nur in der Kombination mit anderen Parametern beeinflussen. Da insgesamt sehr viele Variationsmöglichkeiten bestehen, welche auszutesten sehr viel Zeit bei höchst fraglichem Ergebnis fordern würde, muss eine Priorisierung dieser Parameter gefunden werden, um bei schlechten Ausgangsleistungen keine weiteren Feineinstellungen vornehmen zu müssen. Dafür müssen

Standardeinstellungen gefunden werden, die im weiteren Testlauf stückweise verändert werden. Dieses Vorgehen entspricht durchaus den gemachten Erfahrungen, da ein relativ schlechtes Ergebnis mit solchen Standardparametern praktisch nie signifikant verbessert werden konnte.

Es folgt eine weitere Liste mit den Parametern, diesmal jedoch sortiert nach ihrem Einfluss auf das Ergebnis bzw. die Reihenfolge der durchgeführten Veränderungen. Zusätzlich wird jeweils die Bandbreite der untersuchten Variationen und die Ausgangseinstellung angegeben.

1. **Kodierung der Trainingsdaten**
   Getestet wird sowohl die bereits bekannte Kodierung mit einer expliziten Speicherung von Ass bis Bube sowie eine reine Sortierung der drei Farben.

2. **Umfang der Trainingsdaten**
   Verwendung finden (zufällig erzeugte) Trainingsdaten mit 2.500, 5.000, 10.000, 25.000, 50.000 und teilweise 100.000 Datensätzen. Als Testdaten werden jeweils gleichgroße Datensätze verwendet.

3. **Dämpfung der Lernrate**
   Mit aktivierter Dämpfung werden im Allgemeinen bessere Werte ermittelt, weshalb dies das Standardvorgehen ist.

4. **Anzahl der versteckten Schichten**
   Der vom WE vorgegebene Standard ist (Nummer der Attribute + Nummer der Klassen) / 2. Die Versuche werden mit zehn Schichten begonnen.

5. **Anzahl der Epochen**
   Als Basis werden 50 Epochen verwendet und dann schrittweise erhöht.

6. **Lernrate und Momentum**
   Die benutzte WE-Vorgabe für die Lernrate liegt bei 0,3 und für das Momentum bei 0,2.

Alle erstellten Modelle werden mit den Testdaten überprüft und die vom WE errechneten Fehler angegeben. Die Höhe des Fehlers schlägt sich direkt auf die Bietabweichung und damit auf die Spielstärke nieder. Ein Modell mit einem *relative absolute error* von unter 60 % erreicht im praktischen Einsatz meistens sehr gute Ergebnisse.

Für die vielversprechendsten Modelle werden anschließend reale Spieltests durchgeführt, bei denen das Modell konkret die Stiche für ein gegebenes Kartenblatt vorhersagen muss.

5.4 Eigentliche Bietfunktion

Das mit dem WE erstellte Modell eines neuronalen Netzes kann direkt zur Klassifikation eines gegebenen Kartenblatts verwendet werden. Eine Einbindung der Modelle in eigenen (Java-)Code ist dank der offenen Schnittstellen, die Weka zur Verfügung stellt, mit einem

geringen Aufwand möglich. Nachdem das jeweilige Modell geladen ist, wird aus den aktuellen Spielkarten ein Kartenschlüssel errechnet und dem KNN als Eingabe übergeben. Selbstverständlich muss das Format des Schlüssels identisch mit dem Format der Daten beim Training des KNNs sein. Als Ausgabe erhält man die vorhergesagte Klasse, welche in diesem Fall den zu erwartenden Stichen entspricht.

5.4.1 Bietkorrektur

Wie bereits erläutert, wird die Klasse beim Training des KNNs als numerisch, nicht als nominal deklariert. Deshalb sind die Ausgaben Gleitkommazahlen, welche für ein konkretes Gebot konvertiert werden müssen. Da die Ausgaben des KNNs den Ausgaben der Bietmethode aus dem vorigen Kapitel entsprechen, kann die gleiche Korrektur der Bietwerte erfolgen. Das genaue Vorgehen ist in 4.4.1 beschrieben.

5.4.2 Nullspiel

Auch für das Nullspiel gelten im Wesentlichen die bereits in 4.4.2 ausführlich behandelten Besonderheiten. Für diese Bietfunktion ist deshalb auch eine dedizierte Methode zur Ansage von Null erforderlich.

Da ein KNN lediglich auf ein Problem trainiert werden kann, die Nullansage jedoch ein prinzipiell von der normalen Ansage unabhängiges Problem darstellt, muss zur Umsetzung einer eigenen Nullfunktion ein eigenes KNN erstellt werden. Dieses unterscheidet sich kaum im Aufbau, einzig die zu bestimmende Klasse ist verschieden. Anstatt die mit dem jeweiligen Blatt bzw. dem Datensatz erreichten Stiche wird nur angegeben, ob das Nullspiel gewonnen oder verloren wurde. Das KNN muss ein Blatt demnach nur einer von zwei Klassen zuordnen. Dadurch wird es möglich, ein Netz mit nominaler Klassifikation zu verwenden. In der Praxis unterscheiden sich die Spielergebnisse eines nominalen nicht von einem numerischen Nullansagemodell.

5.5 Ergebnisse

Aus den bereits in 4.5 ausführlich erläuterten Gründen umfassen die folgenden Ergebnisse nur das normale Spiel. Das Nullspiel wird in Abschnitt 5.5.3 separat behandelt.

5.5.1 Klassifikationsleistung

In den folgenden Tabellen gibt die Spalte Datensätze jeweils den Umfang der Trainings- und Testdaten an. D steht für die Dämpfung des KNNs, VS für die Anzahl der versteckten Schichten und Epochen sind die durchgeführten Trainingsläufe. Zusätzlich werden die vom WE berechneten Werte zur Analyse des entstandenen Modells angegeben, dies sind *correlation coefficient* (CC), *mean absolute error* (MAE), *root mean squared error* (RMSE), *relative absolute error* (RAE) und *root relative squared error* (RRSE).

Tabelle 5.2 zeigt die Ergebnisse von Versuchen mit unterschiedlichen Parametern bei identischen Trainingsdaten. Die mit Abstand größte Verbesserung des Ergebnisses wurde

durch die Verwendung der Lernratendämpfung erzielt. Da dieser Einfluss immer positiv ist, wird bei den zukünftigen Versuchen immer mit aktivierter Dämpfung gearbeitet.

Die Anzahl der versteckten Schichten hat keinen ganz so großen Einfluss, in der Praxis hat sich ein Wert um zehn als gut herausgestellt. Die Trainingsepochen spielen eine große Rolle bei KNNs ohne Lernratendämpfung, da durch einen zu großen Wert das Netz zu sehr an die Trainingsdaten angepasst wird, wodurch der Fehler auf den Testdaten wieder steigt. Beim Training mit aktivierter Dämpfung wird dieser Effekt vermieden. Vor allem anfänglich steigt die Klassifikationsleistung mit den Epochen spürbar an. Abhängig von den Trainingsdaten ist ab ca. 150 Epochen jedoch keine große Veränderung mehr zu erkennen.

Die Lernrate sowie das Momentum haben (mit Lernratendämpfung) einen sehr geringen Einfluss auf die Ergebnisse, weshalb sie in den Tabellen nicht extra berücksichtigt wurden. Selbst bei extremen Veränderungen (1,0 statt 0,3 bei der Lernrate oder ein Momentum von 0) ist die Abweichung vom ursprünglichen Zustand sehr klein.

Durch das bereits beschriebene Vorgehen sind in den Tabellen nicht alle möglichen Kombinationen getestet. Vielmehr wurde bei guten Anfangsergebnissen versucht, diese durch eine Optimierung der Parameter weiter zu steigern. Dies deckt zwar nicht alle Fälle ab, ist jedoch ein akzeptabler Kompromiss um gute Einstellungen zu finden. Der komplette (wenn auch beschränkte) Parameterraum könnte nur mit immensem Zeitaufwand ausgetestet werden, ob dabei bessere Ergebnisse gefunden werden könnten ist jedoch fraglich.

Die Resultate von einem Training mit unterschiedlich umfangreichen Trainingsdaten sind in Tabelle 5.3 zu sehen. Die Klassifikationsleistung des KNNs steigt mit der Anzahl der Trainingsdaten an. Die Ergebnisse der Tests mit wenigen Trainingsdaten (2.500 und 5.000) schwanken relativ stark mit der Qualität der Daten. Da diese zufällig generiert werden, empfehlen sich zur Verminderung dieser Effekte eher Modelle, die auf größeren Datenmengen basieren. KNNs mit den gleichen Parametern aber unterschiedlichen Trainingsdaten variieren bei 5.000 Datensätzen z.B. beim RMSE um bis zu 3 Prozentpunkte.

Für einen weiteren Testlauf wurden die Spielkarten nicht vereinfacht, sondern lediglich sortiert an das KNN übergeben (Tabelle 5.4). Die damit erzielten Ergebnisse ähneln den vorangegangen stark. Gute Ergebnisse werden mit 10 versteckten Schichten und ab 100 Epochen erzielt. Interessant sind an den Resultaten dieses Tests vor allem zwei Dinge:

1. Es werden nicht mehr Trainingsdaten für vergleichbare Ergebnisse benötigt.
2. Das jeweils beste Modell liefert beinahe identische Ergebnisse (auch in der praktischen Anwendung, siehe nächster Abschnitt).

Dieses Ergebnis zeigt (neben den bisherigen Vergleichen verschiedener Vereinfachungsstufen), dass die in dieser Arbeit entworfene und verwendete Vereinfachung der Spielkarten kein einschränkender Faktor bezüglich der Bietleistung ist.

Eine exemplarische Lernkurve über 250 Epochen zeigt Abbildung 5.1. Das hierfür verwendete Modell basiert auf 10.000 Datensätzen (sortierte Farben ohne weitere Vereinfachung) mit 10 versteckten Schichten und Lernratendämpfung. Dargestellt ist der RMSE auf den Trainingsdaten und unabhängigen Testdaten mit jeweils 20.000 Datensätzen.

Der Fehler ist auf den Testdaten gegen Ende des Trainings um ca. 0,055 höher als auf den Trainingsdaten. Durch die benutzte Lernratendämpfung gibt es praktisch keine Überanpassung: Auf den Testdaten erreicht der RMSE bei 100 Epochen mit 0,9812 sein Minimum und steigt danach bis zur 250. Epoche lediglich wieder um 0,001. Im gleichen Zeitraum verbessert sich die Leistung auf den Trainingsdaten um 0,0063.

Minimalmodelle

Neben der Suche nach einem Modell mit den besten Klassifikationsleistungen habe ich auch eine Untersuchung von Modellen durchgeführt, die mit extrem wenigen Spielergebnissen der Vereinfachungsstufe *Ass bis Bube* trainiert wurden. Dabei hat sich gezeigt, dass bereits ein auf nur 500 Datensätzen basierendes Modell erstaunlich gute Ergebnisse liefert. Dadurch ist ein Training gegen langsame Gegner und evtl. sogar menschliche Spieler möglich.

In Tabelle 5.5 sind die Ergebnisse für vier Modelle mit je 50, 125, 250 und 500 Trainingsdatensätzen aufgelistet.

5.5.2 Spielstärke

Für die Messungen der Spielstärke wurden drei Modelle mit theoretisch guten Klassifikationsleistungen herangezogen:

1. (M1) **AKDB-5k-e250-h5-d**
   Vereinfachung: Ass bis Bube, Training: 5.000 Datensätze, Epochen: 250, versteckte Schichten: 5, aktivierte Lernratendämpfung

2. (M2) **AKDB-100k-e150-h10-d**
   Vereinfachung: Ass bis Bube, Training: 100.000 Datensätze, Epochen: 150, versteckte Schichten: 10, aktivierte Lernratendämpfung

3. (M3) **Sort-25k-e300-h10-d**
   Vereinfachung: Sortierung der Farben, Training: 25.000 Datensätze, Epochen: 300, versteckte Schichten: 10, aktivierte Lernratendämpfung

Alle drei Bietmodelle mussten mit verschiedenen Bietkorrekturen gegen die statische KI jeweils 5 Millionen Runden spielen. Dabei zeigte sich, dass die drei Modelle allesamt auf sehr hohem Niveau bieten. Die Ergebnisse dieser Testläufe zeigen die Tabellen 5.6, 5.7 und 5.8. Anschließend spielten die drei Modelle mit einer Bietkorrektur von -0,5 gegeneinander. Dabei stellte sich M2 als das beste Modell heraus, wenn auch mit äußerst knappem Vorsprung (Tabelle 5.9).

Da M2 die beste Spielleistung erbracht hat, wurde es für den Test gegen die erfolgreichste Vereinfachungsform der auf Durchschnittswerten basierten Bietfunktion (MH: Vereinfachung Ass bis Dame + Pikbube) verwendet. In Tabelle 5.10 sind die Ergebnisse für zwei Spiele mit je 5 Millionen Runden zu sehen. Im ersten Spiel betrug die Bietkorrektur für beide Methoden -0,5. Hier trug das Weka-Modell knapp den Sieg davon. Für

das zweite Spiel wurden die Bietkorrekturen so angepasst, dass beide Spieler im Endeffekt eine Bietabweichung von genau 0,5 erreichten. Diesen Test gewann das MH-Modell, jedoch nochmals knapper als M2 in der Runde zuvor. Keines der beiden Modelle ist dem jeweils anderen überlegen, das Niveau der Ansagen ist praktisch identisch.

Abschließend musste sich das Weka-Modell M2 noch gegen den GGZ-Spieler beweisen. Die Ergebnisse der Spiele mit je 10.000 Runden und unterschiedlicher Bietkorrekturen zeigt Tabelle 5.11. Dabei sollte das Ergebnis nicht direkt mit den Ergebnissen der Durchschnittsmethode gegen den GGZ-Spieler verglichen werden. Zwar wurden die gleichen Einstellungen verwendet, jedoch sind die statistischen Schwankungen auf Grund der kleinen Rundenzahl relativ groß. Dies ist deutlich an den Ergebnissen der statischen Bietmethode zu sehen, die im Vergleich zu Tabelle 3.1 über 2 Prozentpunkte schlechter abgeschnitten hat. Wie bereits im vorigen Kapitel erläutert, sind bedingt durch die niedrige Geschwindigkeit der GGZ-Spiele der Rundenzahl zeitliche Grenzen gesetzt. Trotz dieser Einschränkung ist deutlich zu erkennen, dass auch bei diesen Testspielen die Variante mit einer Bietkorrektur von -0,5 am erfolgreichsten war.

Minimalmodelle

Die Minimalmodelle müssen sich ebenfalls gegen die statische KI bewähren (Tabelle 5.12). Neben dem Test mit einer Bietkorrektur von -0,5 wurde diese (sofern erforderlich) zusätzlich in 0,05 großen Schritten angepasst, bis eine Abweichung von ca. 0,5 erreicht war.

Bereits das kleinste Modell, welches nur auf 50 Trainingsdatensätzen basiert, erreicht die Bietleistung der statischen Methode. Die nächstgrößeren Modelle mit 125 bzw. 250 Spielergebnissen liegen ungefähr gleichauf. Bereits ab einer Trainingsmenge von 500 Einträgen erreicht das Modell eine Spielleistung von über 60 % und liegt damit nur noch 4,3 % unter dem Ergebnis des mit 5.000 Einträgen zehnmal so großen Modells.

Zusätzlich zeigt Tabelle 5.13 die Ergebnisse der Minimalmodelle gegen das auch in den anderen Testreihen verwendete MH-Modell. Die Resultate fielen wie erwartet aus. Das größte Modell siegte in über 46 % der Spiele und liegt damit etwas unter der Leistung der Durchschnittswertmethode mit der Vereinfachungsstufe *Ass bis König*.

5.5.3 Ergebnisse des Nullspiels

Für das Nullspiel gelten die gleichen Einschränkungen, wie sie bereits sehr ausführlich in 4.5.4 beschrieben wurden.

Da die Entscheidung für oder gegen ein Nullspiel binärer Natur ist, kann ein KNN mit nominaler Klassifikation zum Einsatz kommen. Durch die bisher verwendete, nicht ideale Vereinfachung für das Nullspiel liegt das Augenmerk für ein spezielles KNN zur Nullspielansage auf Modellen mit lediglich sortierender Vereinfachung. Zum Vergleich werden auch die Ergebnisse einiger auf konventionell vereinfachten Daten aufbauender Modelle gezeigt.

Durch die nominale Klassifikation können in den Ergebnistabellen neben den bereits bekannten Fehlern RMSE und RAE noch die Anzahl der korrekt klassifizierten Instanzen

(CCI) sowie Precision und Recall für die Klasse der Nullkarten (Pr 0 und Rc 0) und der Nicht-Nullkarten (ein Stich oder mehr, deshalb Pr 1 und Rc 1) angegeben werden. Bei allen Modellen wurde die Lernratendämpfung benutzt, weshalb dies nicht zusätzlich vermerkt ist.

Tabelle 5.14 zeigt die Ergebnisse bei der Verwendung von nur nach Farben sortierten Trainingsdatensätzen. Die meisten korrekten Klassifizierungen auf den Testdaten gelingt dem auf 50.000 Datensätzen basierenden Modell, wohingegen der RMSE und RAE bei einem nur halb so großen Modell wesentlich niedriger ist. Im Praxistest werden deshalb die folgenden zwei Modelle überprüft:

1. (MN1) **Sort-25k-e100-h5-d**
   Vereinfachung: Farben sortiert, Training: 25.000 Datensätze, Epochen: 100, versteckte Schichten: 5, aktivierte Lernratendämpfung

2. (MN2) **Sort-50k-e100-h5-d**
   Vereinfachung: Farben sortiert, Training: 50.000 Datensätze, Epochen: 100, versteckte Schichten: 5, aktivierte Lernratendämpfung

Ergebnisse unter Verwendung der Vereinfachung von Ass bis Bube zeigt Tabelle 5.15. Wie erwartet sind die Ergebnisse insgesamt etwas schlechter, wobei der CCI-Wert nicht sonderlich von den Werten der vorigen Modelle abweicht. Für den praktischen Einsatz wird folgendes Modell getestet:

3. (MN3) **AKDB-50k-e100-h10-d**
   Vereinfachung: Ass bis Bube, Training: 50.000 Datensätze, Epochen: 100, versteckte Schichten: 10, aktivierte Lernratendämpfung

Als KNN für die normale Bietfunktion wurde für MN1 und MN2 das Modell M3 sowie M2 für MN3 eingesetzt. Alle drei Modelle spielten 5 Millionen Runden gegen zwei Gegner mit statischer Bietmethode. Wie erwartet liegen die Leistungen der beiden ersten Modelle etwas über denen von MN3. Das beste Ergebnis wurde von MN2 erbracht, das sich bereits durch den höchsten CCI auszeichnete.

Insgesamt liegt die Spielleistung mit Nullspiel damit über den Ergebnissen der Durchschnittswertmethode (Tabelle 5.16). Allerdings muss dazu angemerkt werden, dass die KNN-Modelle durch die wesentlich detailliertere Vereinfachung für das Nullspiel besser geeignet sind, während bei den Versuchsreihen der anderen Methode die normale Vereinfachung auch auf das Nullspiel übertragen wurde. Hier ließe sich durch eine separate Vereinfachung für das Nullspiel sicherlich die Spielstärke weiter steigern.

Trotz der Unterschiede spielen beide Methoden insgesamt auf ähnlich hohem Niveau.

Datensätze	D	VS	Epochen	CC	MAE	RMSE	RAE	RRSE
5.000	N	3	10	0,7643	0,8079	1,0123	63,0776 %	64,6573 %
5.000	N	5	5	0,7705	0,7954	0,9994	62,0998 %	63,8315 %
5.000	N	5	10	0,769	0,8	1,007	62,4594 %	62,3197 %
5.000	N	5	25	0,7673	0,8031	1,012	62,6988 %	64,6365 %
5.000	N	5	50	0,7657	0,804	1,0164	62,7709 %	64,9177 %
5.000	N	5	100	0,7594	0,812	1,0276	63,3979 %	65,6326 %
5.000	N	10	10	0,775	0,7897	0,9914	61,6538 %	63,3219 %
5.000	J	1	50	0,7835	0,78	0,9738	60,8943 %	62,1954 %
5.000	J	3	50	0,7846	0,7756	0,9714	60,5506 %	62,0426 %
5.000	J	7	50	0,791	0,7647	0,9587	59,7027 %	61,2355 %
5.000	J	5	10	0,7871	0,7721	0,9674	60,2812 %	61,7907 %
5.000	J	5	25	0,7896	0,7664	0,9608	59,8348 %	61,3648 %
5.000	J	5	50	0,791	0,7643	0,9587	59,6731 %	61,2349 %
5.000	J	5	100	0,7921	0,7625	0,9567	59,5341 %	61,1039 %
5.000	J	5	150	0,7925	0,7616	0,9556	59,4633 %	61,0339 %
5.000	J	5	250	0,7929	0,7608	0,9545	59,4007 %	60,965 %
5.000	J	10	50	0,7915	0,7636	0,9575	59,6192 %	61,1564 %
5.000	J	10	100	0,7924	0,7622	0,956	59,5038 %	61,0591 %
5.000	J	10	150	0,7926	0,7615	0,9553	59,4555 %	61,0126 %
5.000	J	10	250	0,7928	0,761	0,9547	59,4141 %	60,9776 %
5.000	J	20	50	0,7892	0,7666	0,9618	59,8485 %	61,4309 %

Tabelle 5.2: Klassifikationsleistung (Ass bis Bube): Variation der Parameter mit identischen Trainingsdaten

Datensätze	D	VS	Epochen	CC	MAE	RMSE	RAE	RRSE
2.500	J	10	50	0,7914	0,7851	0,9737	60,8621 %	61,1784 %
5.000	J	10	50	0,7915	0,7636	0,9575	59,6192 %	61,1564 %
10.000	J	10	50	0,7847	0,7723	0,9661	61,6074 %	61,9883 %
25.000	J	10	50	0,7871	0,7644	0,9643	60,7063 %	61,7954 %
25.000	J	10	150	0,7887	0,7613	0,9603	60,4577 %	61,5383 %
50.000	J	10	50	0,7946	0,7592	0,9517	59,996 %	60,7957 %
50.000	J	10	150	0,7949	0,7584	0,9505	59,9279 %	60,7177 %
100.000	J	10	50	0,7948	0,7521	0,9465	59,7865 %	60,7018 %
100.000	J	10	150	0,7959	0,7495	0,9442	59,5848 %	60,5525 %

Tabelle 5.3: Klassifikationsleistung (Ass bis Bube): Testreihen mit unterschiedlich umfangreichen Trainingsdaten

Datensätze	D	VS	Epochen	CC	MAE	RMSE	RAE	RRSE
2.500	J	10	50	0,7652	0,7896	0,9973	64,0148 %	64,9548 %
5.000	J	10	50	0,7712	0,8079	1,0089	63,1171 %	63,7934 %
10.000	J	10	50	0,7892	0,7619	0,9553	60,7522 %	61,4471 %
10.000	J	10	150	0,7885	0,7629	0,9568	60,829 %	61,5397 %
25.000	J	5	50	0,788	0,7859	0,9846	62,2188 %	63,3515 %
25.000	J	10	50	0,7946	0,76	0,9565	60,1695 %	61,5403 %
25.000	J	10	100	0,7951	0,7526	0,9468	59,579 %	60,9178 %
25.000	J	10	150	0,7951	0,7513	0,9449	59,4755 %	60,7959 %
25.000	J	10	300	0,7952	0,7505	0,9437	59,4138 %	60,7134 %
25.000	J	15	50	0,7939	0,7601	0,9571	60,1788 %	61,5781 %
25.000	J	15	100	0,7939	0,7547	0,9498	59,7506 %	61,1099 %
25.000	J	15	150	0,7939	0,7535	0,948	59,6531 %	60,9969 %
50.000	J	10	50	0,7883	0,779	0,9783	61,5845 %	62,5834 %

Tabelle 5.4: Klassifikationsleistung (Farben sortiert): Testreihen mit unterschiedlich umfangreichen Trainingsdaten

Datensätze	D	VS	Epochen	CC	MAE	RMSE	RAE	RRSE
50	J	5	50	0,6329	0,9289	1,2114	78,9348 %	78,7165 %
125	J	5	100	0,6885	0,9673	1,2098	72,1298 %	74,7487 %
250	J	5	100	0,7408	0,8619	1,0891	68,432 %	66,8583 %
500	J	5	100	0,7918	0,7707	0,9539	62,7832 %	63,1054 %

Tabelle 5.5: Klassifikationsleistung von Minimalmodellen (Ass bis Bube)

Bietkorrektur	Siege	Abweichung (Betrag)	Stat. Abw. (Betrag)
-0,3	58,39 %	0,30 (0,77)	0,54 (0,94)
-0,4	62,09 %	0,40 (0,80)	0,54 (0,94)
-0,5	62,74 %	0,49 (0,84)	0,54 (0,94)
-0,6	61,61 %	0,59 (0,88)	0,54 (0,94)
-0,7	57,99 %	0,67 (0,93)	0,54 (0,94)

Tabelle 5.6: Spielergebnisse gegen die statische Bietfunktion. Modell M1 (AKDB-5k-e250-h5-d)

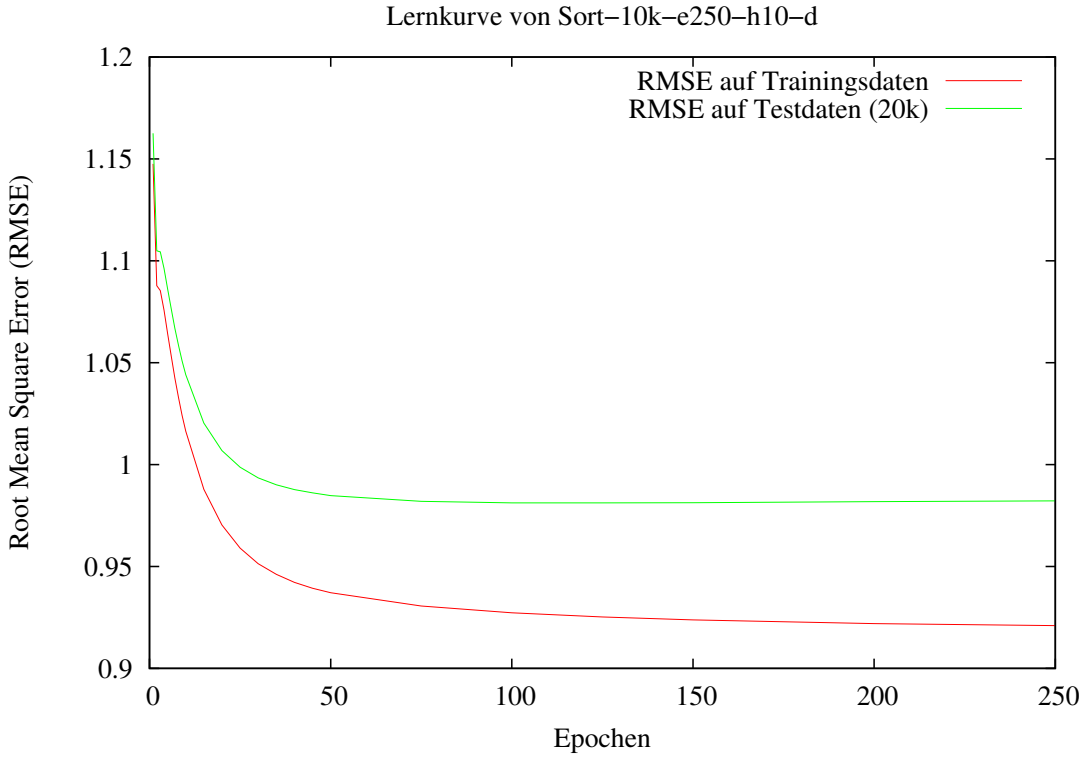

Abbildung 5.1: Lernkurve eines KNN-Modells (sortierte Farben, 10.000 Datensätze) über 250 Epochen

Bietkorrektur	Siege	Abweichung (Betrag)	Stat. Abw. (Betrag)
-0,3	56,42 %	0,25 (0,75)	0,54 (0,94)
-0,4	61,18 %	0,34 (0,77)	0,54 (0,94)
-0,5	63,62 %	0,44 (0,81)	0,54 (0,94)
-0,6	63,23 %	0,53 (0,85)	0,54 (0,94)
-0,7	60,80 %	0,62 (0,89)	0,54 (0,94)

Tabelle 5.7: Spielergebnisse gegen die statische Bietfunktion. Modell M2 (AKDB-100k-e150-h10-d)

Bietkorrektur	Siege	Abweichung (Betrag)	Stat. Abw. (Betrag)
-0,3	56,30 %	0,26 (0,76)	0,54 (0,94)
-0,4	61,22 %	0,36 (0,78)	0,54 (0,94)
-0,5	63,20 %	0,45 (0,82)	0,54 (0,94)
-0,6	63,00 %	0,54 (0,86)	0,54 (0,94)
-0,7	60,28 %	0,64 (0,91)	0,54 (0,94)

Tabelle 5.8: Spielergebnisse gegen die statische Bietfunktion. Modell M3 (Sort-25k-e300-h10-d)

Team 1	Siege	Abweichung	Team 2	Siege	Abweichung
M1	48,77 %	0,49 (0,84)	M2	51,23 %	0,44 (0,81)
M1	49,20 %	0,49 (0,84)	M3	50,80 %	0,45 (0,82)
M2	50,39 %	0,44 (0,81)	M3	49,61 %	0,45 (0,82)

Tabelle 5.9: Spielergebnisse der drei Modelle im direkten Vergleich

Team 1	Siege	BK	Abweichung	Team 2	Siege	BK	Abweichung
M2	50,22 %	-0,5	0,44 (0,81)	MH	49,78 %	-0,5	0,48 (0,82)
M2	49,89 %	-0,57	0,50 (0,83)	MH	50,11 %	-0,52	0,50 (0,83)

Tabelle 5.10: Spielergebnisse des Weka-KNNs gegen die Durchschnittswertfunktion

Bietf.	Bietk.	Siege	Abweichung S1/S3	Abw. S2/S4 (GGZ)
Stat.	-	51,35 %	0,59 (1,02) / 0,64 (1,18)	0,11 (1,25) / 0,06 (1,23)
M2	-0,3	54,99 %	0,23 (0,75) / 0,29 (0,98)	0,21 (1,26) / 0,17 (1,26)
M2	-0,4	59,01 %	0,33 (0,79) / 0,41 (1,01)	0,19 (1,30) / 0,19 (1,22)
M2	-0,5	64,50 %	0,48 (0,86) / 0,52 (1,05)	0,13 (1,23) / 0,06 (1,28)
M2	-0,6	62,98 %	0,58 (0,91) / 0,62 (1,09)	0,11 (1,24) / 0,08 (1,23)
M2	-0,7	63,85 %	0,70 (0,98) / 0,77 (1,18)	0,05 (1,23) / 0,04 (1,20)

Tabelle 5.11: Spielergebnisse gegen den GGZ-Spieler

Datensätze	Bietk.	Siege	Abweichung (Betrag)	Stat. Abw. (Betrag)
50	-0,5	49,90 %	0,61 (0,97)	0,54 (0,94)
50	-0,4	50,76 %	0,52 (0,92)	0,54 (0,94)
125	-0,5	51,15 %	0,37 (0,85)	0,54 (0,94)
125	-0,65	54,18 %	0,52 (0,90)	0,54 (0,94)
250	-0,5	55,46 %	0,51 (0,89)	0,54 (0,94)
500	-0,5	59,56 %	0,45 (0,85)	0,54 (0,94)
500	-0,55	60,04 %	0,49 (0,87)	0,54 (0,94)

Tabelle 5.12: Spielergebnisse von Minimalmodellen (Ass bis Bube) gegen die statische Bietfunktion

Team 1	Siege	BK	Abweichung	Team 2	Siege	BK	Abweichung
50	37,31 %	-0,4	0,52 (0,92)	MH	62,69 %	-0,5	0,48 (0,82)
125	40,57 %	-0,65	0,52 (0,90)	MH	59,43 %	-0,5	0,48 (0,82)
250	41,31 %	-0,5	0,51 (0,89)	MH	58,69 %	-0,5	0,48 (0,82)
500	46,04 %	-0,55	0,50 (0,87)	MH	53,96 %	-0,5	0,48 (0,82)

Tabelle 5.13: Spielergebnisse von Minimalmodellen (Ass bis Bube) gegen die Durchschnittswertfunktion

DS	VS	Epochen	RMSE	RAE	CCI	Pr 0	Rc 0	Pr 1	Rc 1
5k	5	100	0,2799	60,052 %	89,24 %	0,614	0,454	0,922	0,958
10k	5	100	0,2895	61,2915 %	88,29 %	0,617	0,437	0,912	0,956
25k	5	50	0,2758	59,6825 %	89,304 %	0,655	0,492	0,921	0,958
25k	5	100	0,2763	59,2811 %	89,344 %	0,657	0,492	0,921	0,958
25k	5	200	0,2768	59,1538 %	89,34 %	0,657	0,493	0,921	0,958
25k	10	100	0,2792	57,6947 %	89,204 %	0,652	0,483	0,92	0,958
25k	10	200	0,2809	57,5246 %	89,152 %	0,649	0,482	0,919	0,958
50k	5	100	0,2731	60,3573 %	89,686 %	0,68	0,484	0,921	0,963

Tabelle 5.14: Klassifikationsleistung von Nullmodellen (Vereinfachung: Farben sortiert)

DS	VS	Epochen	RMSE	RAE	CCI	Pr 0	Rc 0	Pr 1	Rc 1
5k	5	100	0,2852	66,2034 %	89,12 %	0,698	0,437	0,91	0,968
10k	5	100	0,2795	64,1396 %	89,08 %	0,66	0,464	0,916	0,938
25k	5	100	0,2818	65,1902 %	88,64 %	0,629	0,464	0,916	0,955
50k	5	100	0,2826	64,4447 %	88,678 %	0,659	0,417	0,909	0,964
50k	10	100	0,2813	62,9714 %	88,856 %	0,667	0,428	0,911	0,965
50k	5	200	0,2824	64,5514 %	88,712 %	0,657	0,426	0,91	0,963

Tabelle 5.15: Klassifikationsleistung von Nullmodellen (Vereinfachung: Ass bis Bube)

Team 1	Siege	Nullspiele (Siege)	Team 2	Siege	Nullspiele (Siege)
MN1	60,87 %	1.029.038 (68,01 %)	Stat.	39,13 %	1.171.174 (61,56 %)
MN2	61,46 %	1.021.163 (69,25 %)	Stat.	38,54 %	1.172.392 (61,63 %)
MN3	59,84 %	953.467 (67,50 %)	Stat.	40,16 %	1.171.150 (61,34 %)

Tabelle 5.16: Spielergebnisse von Nullmodellen gegen die statische Bietfunktion

6 Zusammenfassung

In dieser Diplomarbeit wurde das Kartenspiel Spades im Allgemeinen und besonders bezüglich seiner Bietphase betrachtet, mit anderen Spielen verglichen und der aktuelle Stand der Forschung im Bereich der Spiel-KI für ähnliche Spiele aufgezeigt. Neben der Entwicklung eines künstlichen Spielers inkl. einer statischen Bietmethode zu Vergleichszwecken wurden zwei unterschiedliche lernfähige Ansätze zur Verbesserung der Bietphase vorgestellt und ausführlich getestet.

Der erste Ansatz basiert auf der Speicherung umfangreicher Spielergebnisse nebst zugehörigen (vereinfachten) Spielkarten in geeigneten Datenstrukturen und das Erzeugen eines Gebots aus den jeweiligen Durchschnittswerten. Die zweite Methode verwendet ein mit ebenfalls realen Spielergebnissen trainiertes künstlichen neuronales Netz (KNN) und damit einen echten Lernalgorithmus zur Voraussage der Stiche zu einem Kartenblatt.

In umfassenden Testreihen wurde gezeigt, dass beide Methoden die Bietleistung (auch gegen andere Gegner) deutlich erhöhen können. Dabei hat sich kein Verfahren als dem anderen überlegen herausgestellt, die Spielstärke beider Lösungen schwankt innerhalb der Messtoleranz und kann somit als praktisch identisch eingestuft werden. Trotzdem haben beide Ansätze jeweils spezifische Vor- und Nachteile und sind daher für unterschiedliche Einsatzzwecke geeignet.

Die auf Durchschnittswerten basierende Bietfunktion ist bei der Verwendung von Hashtabellen als Datenspeicher sehr schnell, erreicht hohe Leistungen jedoch erst nach einem umfangreichen Training, wie es lediglich gegen einen künstlichen Gegner möglich ist. Vorteilhaft ist hingegen die Art des Trainings, da dieses während des normalen Spielablaufs erfolgt. Der Spieler verbessert dadurch beständig sein Spiel, bis die Maximalleistung erreicht ist. Durch den Einsatz einer alternativen Bietfunktion als Rückfallsicherung bei unzureichenden Spielerfahrungen liegt die Leistung von Beginn an auf hohem Niveau. Die gespeicherten Datentypen können um weitere Informationen ergänzt werden, dadurch eignet sich eine gut befüllte Hashtabelle ideal für verschiedene Auswertungen der Spieleigenschaften.

Der größte Vorteil der KNN-Lösung liegt in der geringen Anzahl benötigter Trainingsdaten. Zwar verbessert sich die Spielleistung auch hier noch mit der Anzahl der Trainingsdatensätze, jedoch kann (bei guten Ausgangsdaten) bereits mit nur 500 Datensätzen ein durchaus gut spielendes und performantes KNN-Modell erstellt werden. Besonders bei großen Modellen, die ohne weitgehende Vereinfachung der Spielkarten erstellt wurden, liegt die Ausführungsgeschwindigkeit jedoch deutlich unterhalb der Performanz des ersten Ansatzes. Dies ist allerdings nur beim lokalen Spiel gegen künstliche Gegner von Belang. Der Nachteil der KNNs ist das umständliche Training mit externen Werkzeugen sowie die nicht mögliche Speicherung zusätzlicher Spieldaten zu Auswertungszwecken.

Ein Sonderfall stellt in beiden Fällen Sonderspiel Null dar. Nur bei diesem Spiel ändert

sich wegen des Gebots die Spielweise aller Spieler, so dass die Ergebnisse sehr stark von der eigentlichen Spiel-KI abhängen. Zusätzlich ist die Bepunktung eines Nullspiels so hoch, dass die Leistung der jeweiligen Null-KI das gesamte Spielergebnis dominiert. Im Gegensatz dazu ist die Erkennung von Spielkarten mit der Möglichkeit eines Nullspiels nicht weiter kompliziert. Aus diesem Grund wurde das Nullspiel in dieser Arbeit zwar ebenfalls erfolgreich untersucht, wegen seines verfälschenden Einflusses jedoch nicht bei den Spielstärketestläufen berücksichtigt sondern separat behandelt und ausgewertet.

Obwohl die Entwicklung einer Spiellogik für die eigentliche Spielphase nicht im Mittelpunkt der Arbeit stand, ist das statische, auf Heuristiken basierende Ergebnis von der Leistung durchaus mit anderen Spades-KIs vergleichbar. Mit der statischen Bietmethode wird bereits die Spielstärke des GGZ-Spielers übertroffen.

Insgesamt wurde das Ziel einer Verbesserung der Bietleistung durch den Einsatz lernfähiger Methoden durchaus zufriedenstellend erreicht. Auch die im Verlauf der Planung getroffenen Annahmen zur Vereinfachung der Spielkarten haben sich (durch den direkten Vergleich entsprechender KNN-Modelle) als vernünftig herausgestellt.

A Entwicklungsdetails

A.1 Erstellung der Simulationsumgebung

In diesem Abschnitt werden kurz die zur Erstellung aller Programmteile verwendeten Applikationen vorgestellt.

A.1.1 Java

Java ist eine moderne, objektorientierte und auch weit verbreitete Programmiersprache. Durch die Verwendung von Bytecode und die anschließende Ausführung in einer virtuellen Maschine sind damit erstellte Anwendungen plattformunabhängig, allerdings nicht immer sehr performant.

Die Wahl auf Java als Programmiersprache ermöglichte die problemlose Integration von Weka in die Simulation, zusätzlich konnten aus dem GGZ-Javaclient Klassen zur Kommunikation mit dem (in C geschriebenen) GGZ-Server benutzt werden.

A.1.2 BEA JRockit

Als virtuelle Maschine kam in der Arbeit JRockit von BEA zum Einsatz. Durch ihren leistungsfähigen Just-in-Time-Compiler konnte damit im Vergleich zur Java-VM von Sun Microsystems die Ausführungsgeschwindigkeit deutlich gesteigert werden.

A.1.3 Eclipse SDK

Eclipse ist eine quelloffene Entwicklungsumgebung zur komfortablen Erstellung von Quelltexten jeglicher Art. Seine Ursprünge liegen in der Javaentwicklung, dadurch sind für diesen Zweck viele ausgereifte Werkzeuge vorhanden. Durch Erweiterungen kann Eclipse für viele verschiedene Programmier-, Skript- und Beschreibungssprachen erweitert werden.

A.1.4 Betriebssystem und Hardware

Als Betriebssystem kam Gentoo Linux (32 Bit) auf einem Intel Core 2 Quad Q6600 (2,4 GHz) mit 2 GB Arbeitsspeicher zum Einsatz. Alle erstellten Programme sind ohne Einschränkung unter Microsoft Windows lauffähig, ebenso auf jedem anderen Betriebssystem, für das eine Java-Laufzeitumgebung existiert.

A.1.5 Lizenz

Alle Quelltexte meiner Arbeit stehen, bedingt durch die Verwendung von Weka- und GGZ-Programmteilen, ebenfalls unter der GNU General Public License.

A.2 Kommunikation mit dem GGZ-Server

Die Verständigung eines Clients mit dem GGZ-Server ist in mehrere Phasen eingeteilt. Bei der ersten Verbindungsaufnahme wird eine TCP/IP-Verbindung zum Server aufgebaut, über welche sich der (Kern-)Client über ein XML-Protokoll anmeldet. Über XML werden auch sämtliche weitere Stausmeldungen und Änderungen übertragen, z.B. das Betreten eines Raums oder das Eröffnen eines Spiels. Diese Teile des Clients und des Servers sind nicht spielspezifisch.

Nachdem alle Voraussetzungen zum eigentlichen Spielstart erfüllt sind, wird die Verbindung sowohl vom Server als auch vom Client an einen spielspezifischen Teil übergeben, die anschließend mit einem binären Protokoll kommunizieren. Nicht jeder Client muss dabei alle Spiele (und alle Protokolle) unterstützen, ebensowenig muss der Server alle Spiele anbieten.

Unabhängig davon bleibt die ursprüngliche Verbindung zwischen Server und Client weiterhin bestehen, der Client wird darüber z.B. über die Nachrichten von anderen Spielern oder anderweitigen Veränderungen benachrichtigt. Auch die letztendliche Abmeldung vom Server geschieht über diese Verbindung.

Für ausführliche Details empfehle ich die Dokumentation des Projekts unter [6].

A.3 Textsatz

A.3.1 LaTeX

Latex ist ein im wissenschaftlichen Umfeld weit verbreitetes Textsatzsystem, das Tex um zusätzliche Makros erweitert. Es ist frei verfügbar für viele Betriebssysteme und zeichnet sich vor allem durch seine Stabilität aus.

A.3.2 Kile

Kile ist eine Entwicklungsumgebung für Latex und wurde speziell für KDE entwickelt. Durch seine Syntaxhervorhebung und Autovervollständigung für Latexbefehle sowie Makros zur schnellen Erstellung und Betrachtung des kompilierten Textes erleichtert es den Umgang mit Latex deutlich.

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[1] Amit, Asaf und Markovitch, Shaul: *Learning to Bid in Bridge.* Machine Learning, 63(3):287–327, 2006.

[2] Fong, Jason: *Developing an Artificial Intelligence for Whist.* University of California, Computer Science, Los Angeles, 2005.

[3] Frank, Ian und Basin, David A.: *Search in Games with Incomplete Information: A Case Study Using Bridge Card Play.* Artificial Intelligence, 100(1-2):87–123, 1998.

[4] Frank, Ian; Basin, David A. und Matsubara, Hitoshi: *Finding Optimal Strategies for Imperfect Information Games.* In: *AAAI/IAAI*, Seiten 500–507, 1998.

[5] Gerhardt, Gunter: *XSkat.* http://www.xskat.de/xskat-latest-de.html, 2007. [Online; Stand 19. November 2007].

[6] *GGZ Gaming Zone.* http://www.ggzgamingzone.org, 2007. [Online; Stand 17. Juli 2007].

[7] Ginsberg, Matthew L.: *GIB: Steps Toward an Expert-Level Bridge-Playing Program.* In: *IJCAI*, Seiten 584–593, 1999.

[8] Hiszpanski, Chris: *SARC - Spades Artificially Reasoning Computer.* http://sarc.sourceforge.net, 2007. [Online; Stand 17. Juli 2007].

[9] *Internationale Skatordnung (Deutscher Skatverband e.V.).* http://www.dskv.de/pages/Skatgericht/ISkO.php, 2007. [Online; Stand 20. November 2007].

[10] Kupferschmid, Sebastian: *Entwicklung eines Double-Dummy Skat Solvers mit einer Anwendung für verdeckte Skatspiele.* Diplomarbeit, Albert-Ludwigs-Universität Freiburg, Juli 2003.

[11] Kuvayev, Leo: *Learning to Play Hearts.* In: *AAAI/IAAI*, Seite 836, 1997.

[12] Russell, Stuart und Norvig, Peter: *Künstliche Intelligenz. Ein moderner Ansatz*, Band 2. Pearson Studium, 2004.

[13] Schaeffer, Jonathan et al.: *Checkers Is Solved.* Science, 14. September 2007. DOI: 10.1126/science.1144079, Seiten 1518–1522.

[14] Schneiderman, John: *KardsGT.* http://kardsgt.nongnu.org, 2007. [Online; Stand 17. Juli 2007].

[15] Schäfer, Jan: *Monte Carlo Simulation im Skat.* Bakkalaureatsarbeit, Otto-von-Guericke-Universität Magdeburg, 2005.

[16] Smith, Stephen J. J.; Nau, Dana S. und Throop, Thomas A.: *Success in Spades: Using AI Planning Techniques to Win the World Championship of Computer Bridge.* In: *AAAI/IAAI*, Seiten 1079–1086, 1998.

[17] Wikipedia: *Artificial neural network.* http://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=174330308, 2007. [Online; Stand 30. November 2007].

[18] Wikipedia: *Bridge scoring.* http://en.wikipedia.org/w/index.php?title=Bridge_scoring&oldid=131851716, 2007. [Online; Stand 20. November 2007].

[19] Wikipedia: *Hearts.* http://de.wikipedia.org/w/index.php?title=Hearts&oldid=38632741, 2007. [Online; Stand 20. November 2007].

[20] Wikipedia: *Kontrakt-Bridge.* http://de.wikipedia.org/w/index.php?title=Kontrakt-Bridge&oldid=27817339, 2007. [Online; Stand 20. November 2007].

[21] Wikipedia: *Oh Hell.* http://en.wikipedia.org/w/index.php?title=Oh_Hell&oldid=171053454, 2007. [Online; Stand 20. November 2007].

[22] Wikipedia: *Skat.* http://de.wikipedia.org/w/index.php?title=Skat&oldid=38988172, 2007. [Online; Stand 20. November 2007].

[23] Wikipedia: *Spades.* http://de.wikipedia.org/w/index.php?title=Spades&oldid=32355289, 2007. [Online; Stand 17. Juli 2007].

[24] Witten, Ian H. und Frank, Eibe: *Data Mining: Practical machine learning tools and techniques (2nd Edition).* Morgan Kaufmann, San Francisco, 2005. Verwendete Softwareversion (Weka): 3.4.10.