

Bridge Baron - Der Computer als Gegner

Andreas Fröhlich

Zusammenfassung

Künstliche Intelligenz bei Bridge - Im Vergleich zu anderen Gebieten ein verhältnismäßig schwieriges Unterfangen, da die üblichen Brute-Force-Methoden daran scheitern, dass die Karten der Gegner nicht bekannt sind. Bridge Baron tastet sich deshalb aus einer anderen Richtung an das Problem heran und versucht mit Hilfe von *HTN*-Planung sich dem Spiel eher so zu nähern, wie es ein menschlicher Spieler versuchen würde. Es wird betrachtet, was genau man unter *HTN*-Planung versteht, wie dieser Ansatz in Bridge Baron implementiert wurde und wie er zu bewerten ist.

„Bridge ist ein phantastisches Spiel, weil es von Strategie und Partnerschafts-Vertrauen lebt. Man muss fähig sein schnell zu denken und zu reagieren.“ – Bill Gates

Inhaltsverzeichnis

1	Einleitung	3
2	Bridge	3
2.1	Regeln	3
2.2	Strategien	4
3	KI bei Bridge	6
3.1	Probleme	6
3.2	HTN	7
3.3	Tignum 2	8
3.4	Ergebnisse	10
4	Weitere Ansätze	11
4.1	Vollständige Suche	11
4.2	Monte Carlo Technik	11
5	Zusammenfassung	12
	Literatur	13

1 Einleitung

Im Laufe der letzten Jahre hat sich die künstliche Intelligenz bei Computerspielen stark verbessert und ist dadurch auch den Herausforderungen, die der Mensch sucht, gerecht geworden. Bei Schach oder ähnlichen Spielen gibt es bereits seit langer Zeit Programme, die sich auch mit den besten menschlichen Spielern messen können. Andere Spiele beherrscht der Computer sogar besser, als es einem Menschen je möglich wäre. In diese Kategorie fallen zum Beispiel Othello und Dame. Anders hingegen verhält es sich bei Bridge, wie auch bei vielen anderen, ähnlichen Kartenspielen. Dort können die meisten Computer-Programme gerade einmal mit einem eher durchschnittlich guten menschlichen Spieler mithalten. Im Folgenden wird beschrieben weshalb dem so ist und mit welchen Ansätzen versucht wird, den Computer-Gegner dennoch stärker zu machen. Bei einem wichtigen Ansatz, der auch, jedoch keinesfalls ausschließlich, bei Bridge Baron Verwendung findet, handelt es sich um die sogenannte, *HTN-Planung* (*Hierarchical Task Network* - Planung).

2 Bridge

Zum besseren Verständnis der Ansatzpunkte und Funktionsweise der künstlichen Intelligenz bei Bridge-Programmen werden in diesem Abschnitt die grundsätzlichen Bridge-Regeln erklärt und einige elementare Strategien angeschnitten. Ein ausführlicheres Regelwerk und eine komplette Anleitung zu Bridge, die auch als Quelle für diesen Abschnitt diente, findet sich unter [Bri03].

2.1 Regeln

Bridge besteht im wesentlichen aus zwei Bestandteilen: Dem Reizen und dem eigentlichen Spiel. Verwendet wird ein gewöhnliches Kartendeck mit 52 Karten, bestehend aus vier verschiedenen Farben (\spadesuit Pik, \heartsuit Coeur, \diamondsuit Karo, \clubsuit Treff). Es spielen jeweils die beiden gegenüberstehenden Personen zusammen. Üblicherweise bezeichnet man die Spieler auch nach den Himmelsrichtungen. Es spielen also Nord und Süd, sowie West und Ost miteinander.

Zu Beginn werden alle Karten unter den Spielern ausgeteilt. Anschließend können die einzelnen Spieler reizen. Ein Gebot besteht dabei aus einer bestimmten Anzahl von Stichen, die der Spieler anschließend machen muss und einer Farbe, welche er als Trumpf vorschlägt. Alternativ kann er auch ein trumpfloses Spiel anmelden. Will kein Spieler mehr ein weiteres Gebot abge-

ben, so wird das Gebot mit der höchsten Anzahl an Stichen gültig und das eigentliche Spiel beginnt.

Derjenige Spieler, der das höchste Gebot abgegeben hat (oder auch sein Partner, falls dieser bereits vorher ein Gebot in der gleichen Farbe abgegeben hat), wird zum sogenannten *Alleinspieler*, sein Partner zum *Dummy*. Die beiden Gegenspieler werden Verteidiger genannt. Die erste Karte legt derjenige, der links vom *Alleinspieler* sitzt. Als zweites ist somit der *Dummy* an der Reihe. Die vier Spieler legen nacheinander im Uhrzeigersinn je eine Karte. Dabei muss, jeder Spieler, falls möglich, Farbe zugeben. Das heißt eine Karte derjenigen Farbe legen, die durch den ersten Spieler angespielt wurde. Besitzt der Spieler keine Karte dieser Farbe, so steht es ihm frei eine beliebige Karte anderer Farbe oder einen Trumpf zu spielen. Haben alle Spieler eine Karte gelegt, so sticht derjenige Spieler, der die höchste Karte in der entsprechenden Farbe ausgespielt hat, falls kein Trumpf gespielt wurde. In diesem Fall geht der Stich an den Spieler, der den höchsten Trumpf gelegt hat.

Sobald der *Dummy* zum ersten Mal an der Reihe ist, deckt er alle seine Karten auf. Der *Alleinspieler* übernimmt ab diesem Zeitpunkt das Spiel für ihn. Diese Besonderheit ist gewissermaßen das zentrale taktische Element von Bridge, auf welches später noch genauer eingegangen wird.

Ein Spiel ist vorbei, sobald kein Spieler mehr eine Karte auf der Hand hat. Danach werden die Stiche gezählt und es wird überprüft, ob der *Alleinspieler* tatsächlich die angekündigte Anzahl an Stichen oder mehr machen konnte. Falls ja, zählt das Spiel als gewonnen, ansonsten als verloren.

2.2 Strategien

Wie bereits erwähnt stellt das Zusammenspiel zwischen *Alleinspieler* und *Dummy* ein zentrales taktisches Element von Bridge dar. Dabei versucht der *Alleinspieler* durch geschicktes Anwenden diverser Strategien möglichst viele Stiche zu machen. An dieser Stelle zum besseren Überblick einige, jedoch längst nicht alle, möglichen Varianten, auf die sowohl der *Alleinspieler* als auch die Verteidiger bei ihrem Spiel zurückgreifen können.

Schnitt (engl. *Finesse*):

Der Spieler eröffnet mit einer Farbe, in der sein Partner zwei hohe, jedoch nicht aufeinanderfolgende Karten auf der Hand hat. Dadurch ist es unter gewissen Umständen möglich, dass beide Karten stechen, obwohl die Karte dazwischen im Besitz des Gegners ist. Beispiel: Der Partner des Ausspielers besitzt Dame und As, der Gegner vor ihm den König der entsprechenden Farbe. Spielt er diesen, kann ihn der Partner des Ausspielers mit dem As überstechen und die Dame ist die höchste Karte im Spiel. Behält er den König auf der Hand, so

kann der Partner direkt mit der Dame stechen.

Hochspielen (engl. *Cash out*):

Es werden direkt die höchsten Karten einer Farbe gespielt mit der Absicht sichere Stiche zu machen.

Überkreuz-Schnappen (engl. *Cross-Ruffing*):

Alleinspieler und *Dummy* bzw. die beiden Verteidiger spielen jeweils abwechselnd eine Farbe, die ihr Partner nicht auf der Hand hat. Der Partner schnappt sich daraufhin den Stich mit einem Trumpf. Dadurch können auch Farben, in denen die hohen Karten beim Gegner sind, Stiche bringen.

Abducken:

Ein möglicher Stich wird gezielt dem Gegner überlassen, um dadurch einen taktischen Vorteil zu erlangen.

Abwerfen:

Der Spieler hat keine Karte der angespielten Farbe und wirft stattdessen aus taktischen Gründen eine Karte ab, mit der er vermutlich keinen Stich machen wird.

Etablieren einer Farbe:

Geschicktes Spielen von verschiedenen Farben oder auch Trumpf wird dazu verwendet, es auch niedrigeren Karten einer Farbe zu ermöglichen einen Stich zu machen weil zum entsprechenden Zeitpunkt alle höheren Karten bereits gespielt wurden.

Trümpfe ziehen:

Durch mehrmaliges Anspielen von Trumpf wird der Gegner trumpfflos gemacht und damit daran gehindert sich im späteren Verlauf des Spielen einen Stich zu schnappen.

Übergang :

Der Spieler spielt eine Farbe, in der sein Partner stechen kann, um es diesem zu ermöglichen als nächstes auszuspielen.

Erkundungsausspiel:

Die Intention des Ausspielers ist es dabei, durch die Reaktion des Gegners Informationen über dessen Blatt zu bekommen.

Darüber hinaus gibt es noch eine Vielzahl verschiedener Varianten, Abwandlungen und Erweiterungen, der soeben beschriebenen Strategien (Beidseitiger Schnitt, Vermeidungsschnitt, Schnapper mit Abwurf, etc.), die im Detail den Rahmen dieser kurzen Erklärung sprengen würden. Jedoch sollte bereits aus den hier beschriebenen Techniken deutlich geworden sein, dass es sich bei Bridge um ein Spiel handelt, in dem es wesentlich ist, die existierenden Stra-

tegien möglichst geschickt anzuwenden um zu gewinnen. Auf ein paar davon wird später noch etwas genauer eingegangen.

3 KI bei Bridge

Während der letzte Abschnitt lediglich einen grundsätzlichen Überblick über Bridge an sich geben sollte, beschäftigt sich dieser insbesondere mit den Problemen der künstlichen Intelligenz und dem Lösungsansatz, der bei Bridge Baron gewählt wurde sowie dessen Hintergründe.

3.1 Probleme

Es stellt sich zunächst die Frage, warum es bisher nicht gelungen ist einen spielstarken Computergegner für Bridge zu entwickeln, der auch mit den besten menschlichen Spielern mithalten kann. Schließlich wirkt Bridge für einen Laien wohl kaum komplizierter als zum Beispiel Schach, für welches dies ja bereits gelungen ist.

Das Problem, mit dem man sich bei Bridge konfrontiert sieht, genauso wie bei vielen anderen Kartenspielen, ist anderer Natur. Bei den meisten bisherigen Ansätzen handelt es sich nämlich um sogenannte Brute-Force-Varianten. Das heißt, das Programm berechnet schlicht und einfach alle Züge, die im momentanen Zustand des Spieles möglich sind und wiederum alle möglichen Folgezüge des Gegners und wiederum alle möglichen eigenen Zügen usw.

Dazu wird im Wesentlichen die sogenannte Minimax-Formel verwendet:

$$\textit{minimax}(p) = \begin{cases} \text{our payoff at node } p & \text{if } p \text{ is a terminal node} \\ \max\{\textit{minimax}(q) : q \text{ is a child of } p\} & \text{if it is our move at the node } p \\ \min\{\textit{minimax}(q) : q \text{ is a child of } p\} & \text{if it is our opponent's move at the node } p \end{cases}$$

Abbildung 1. Minimax-Formel

Bei anderen Spielen, wie eben das schon des öfteren erwähnte Schach, ist dieser Ansatz durchaus praktikabel und führt, wie sich gezeigt hat, auch zu annehmbaren Ergebnissen. Dabei werden dort heutzutage natürlich auch zahlreiche weitere Methoden zur Optimierung angewendet. Der Unterschied bei Bridge jedoch ist, dass es sich dabei um ein Spiel handelt, bei dem der Computer nicht den kompletten Zustand der momentanen Partie kennt. Dies ist naheliegend, denn schließlich weiss ein Spieler im Normalfall nicht, welche Karten

seine Gegner auf der Hand haben. Durch diese Tatsache muss der Computer nicht nur alle möglichen Spielzüge, die ein Gegner in einem gegebenen Zustand machen könnte, berechnen sondern dies für alle möglichen Verteilungen der Karten durchführen. Durch diesen Unsicherheitsfaktor steigt der dazu benötigte Rechenaufwand allerdings immens.

Dazu kommt, dass bei Bridge nicht mehrere Minuten für die Berechnung eines Zuges zur Verfügung stehen, sondern ein komplettes Spiel in verhältnismäßig kurzer Zeit, für gewöhnlich in fünf bis zehn Minuten, abläuft. Dadurch können Computer schlicht und einfach nicht genügend Berechnungen durchführen, um zu einem akzeptablen Ergebnis, das sich auch im Spiel bewährt, zu gelangen. Als Folge davon wurden diverse andere Ansätze versucht, um dieses dennoch zu erreichen.

3.2 HTN

Der Ansatz, den Bridge Baron dazu verwendet, ist eine leichte Abwandlung der sogenannten *Hierarchical Task Network* (*HTN*) Planung. Dieser Ansatz bietet sich deshalb an, da es sich bei Bridge, wie bereits weiter oben veranschaulicht, um ein Spiel handelt, bei dem die taktische Planung und das Ausspielen vorhandener Strategien eine zentrale Bedeutung hat.

Der *HTN*-Begriff bezeichnet das Vorgehen mögliche Tätigkeiten oder auch Aufgaben (engl. *Tasks*) jeweils in eine endliche Anzahl weiterer, untergeordneter Aufgaben zu untergliedern, so lange bis nur noch elementare Aufgaben vorhanden sind. Diese elementaren Aufgaben können dann direkt ausgeführt werden. Dabei besitzt jede Aufgabe bestimmte Vorbedingungen, die erfüllt sein müssen, damit eine Ausführung überhaupt möglich ist.

Ein Beispiel, hier nicht in Verbindung mit Bridge, ist in Abb. 2 veranschaulicht.

HTN entspricht damit eher der Vorgehensweise, die auch ein Mensch anwenden würde, wenn er eine gewisse Tätigkeit plant. Damit entspricht der Ansatz auch der Art, auf die ein menschlicher Spieler an Bridge, wie auch an viele andere Spiele herangehen würde. Der Vorteil eines solchen Vorgehens besteht darin, dass nicht mehr alle möglichen Karten, die gespielt werden können, sondern nur die verschiedenen, auf eine Situation anwendbaren, Strategien betrachtet werden müssen. Da die Anzahl dieser Strategien im Allgemeinen deutlich geringer als die aller möglichen Karten ist, verringert sich dadurch auch die Menge der zu überprüfenden Knoten bei der Berechnung drastisch. Bei Spielen wie Schach, in denen jeweils der komplette Zustand des Spiels bekannt ist, hat sich *HTN* bisher jedoch nicht bewährt.

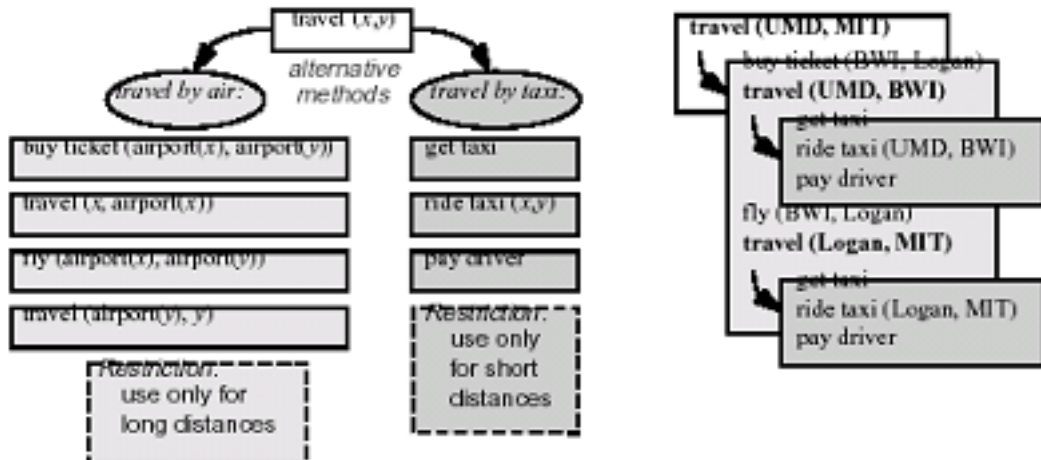


Abbildung 2. Beispiel einer HTN-Zerlegung. Die Methode `travel(x,y)` kann auf zwei Arten zerlegt werden. Jeder Fall hat eine Vorbedingung.

	Brute Force	HTN-Planung
Maximal	ca. 5.6×10^{44} Knoten	ca. 305'000 Knoten
Durchschnittlich	ca. 2.3×10^{24} Knoten	ca. 26'000 Knoten

Abbildung 3. Reduzierung der Knotenzahl durch HTN-Planung.

Es existieren noch einige weitere Bereiche, in denen *HTN*-Planung bereits durchaus erfolgreich angewendet wird. Ein Beispiel dafür ist die Herstellung von Mikrowellen-Übertragungs-Modulen. Dort wird sogar teilweise Programm-Code verwendet, der identisch zum im nächsten Abschnitt beschriebenen Tignum 2-Code ist. Dies sollte jedoch nur veranschaulichen, dass Bridge keineswegs die einzige Anwendung von *HTN* ist und uns, an dieser Stelle, nicht weiter im Detail beschäftigen. (Siehe auch [OP91],[SIPE99],[UMCP].)

3.3 Tignum 2

Das bei Bridge Baron verwendete Tignum 2 ist die Implementierung des im vorherigen Kapitel beschriebenen *HTN* bei Bridge. Es weist allerdings einige, für dieses Spiel notwendige, leichte Abwandlungen von *HTN* auf. Hinzugefügt wurde die Möglichkeit mehrere Spieler zu verwalten und die Fähigkeit mit Wahrscheinlichkeiten zu arbeiten, da bei Bridge nur selten eine sichere Aussage über das Blatt des Gegners möglich ist. Eine Einschränkung hingegen wurde in der Abarbeitung der Aufgaben gemacht. Verläuft diese bei *HTN* nur teilweise geordnet, so läuft sie bei der Implementierung in Tignum 2 komplett geordnet ab. Damit werden einige Probleme umgangen, die dadurch auftreten könnten, dass zunächst ein Schritt ausgeführt wird, in dem eine Karte gespielt wird und danach die Zuordnung dieser Karte zu einem Spieler vorgenommen

werden soll. Letztendlich sind die Vorteile, die durch eine teilweise geordnete Abarbeitung entstehen würden bei Bridge jedoch so gering, dass sich der Aufwand dafür nicht lohnt.

Abb. 4 zeigt ein Beispiel, wie *HTN* für Bridge in Tignum 2 implementiert wurde. Der *Alleinspieler* könnte zum Beispiel einen Schnitt versuchen, da die dazu nötigen Gegebenheiten erfüllt sind. Die möglichen Abläufe werden wie folgt unterteilt:

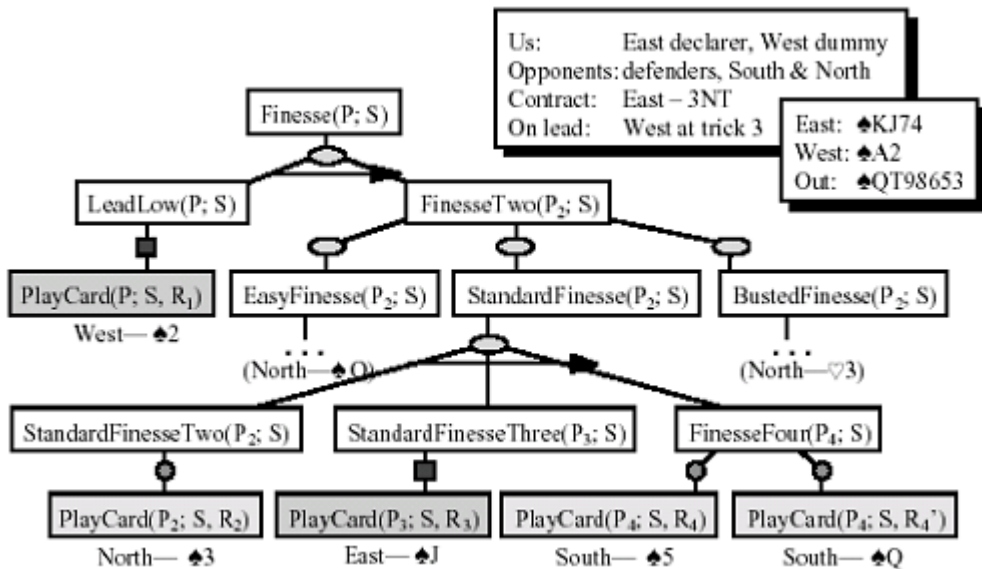


Abbildung 4. Aufbau der Schnitt-Methode.

In Abb. 4 ist insbesondere zweierlei zu erkennen: Die Erweiterung gegenüber gewöhnlichem *HTN* dadurch, dass auch die möglichen Züge des Gegners als Knoten im Baum eingetragen sind und nicht nur die eines Spielers. Zudem fällt auf, dass es nur eine geringe Anzahl von Knoten gibt, da es für das Gelingen des Schnitts uninteressant ist, ob der Gegner zum Beispiel Pik 3, Pik 4, oder gar Pik 10 spielt. In allen Fällen wirkt sich dies nicht auf den weiteren Spielverlauf aus. Da sich Pik As, König und Bube beim *Alleinspieler* und beim *Dummy* befinden wirken sich einzig die beiden Varianten „Der Gegner spielt Pik Dame.“ und „Der Gegner spielt eine andere Farbe.“ aus. In beiden Fällen spielt der Partner natürlich den König. Im ersten um den Stich zu machen und im zweiten, weil er nun mit Sicherheit weiss, dass sich die Dame beim Spieler nach ihm befindet und der Schnitt damit also sicher scheitern würde.

Auf diese Weise wird nun ein Baum mit allen möglichen Strategien aufgebaut. Welche davon anwendbar sind, wird jeweils durch die entsprechenden Vorbedingungen entschieden. Wie in Abb. 4 ersichtlich, enthält dieser Baum nur eine vergleichsweise geringe Anzahl an Knoten und kann demzufolge komplett durchlaufen und ausgewertet werden.

Zu beachten ist jedoch, dass die Auswertung hier nicht wie sonst oft üblich nach dem Minimax-Prinzip durchgeführt werden kann. Der Grund dafür ist auch hier wieder die Tatsache, dass dem jeweiligen Spieler nicht bekannt ist, welche Karten der Gegner auf der Hand hat. Die Auswertung bei den Knoten, die den Zügen des Gegners entsprechen, muss also anders erfolgen. Dazu wird jedem möglichen Fall, der sich unterschiedlich auf das Fortführen oder den Ausgang der Strategie auswirkt, eine Wahrscheinlichkeit zugeordnet. Der Knoten selbst bekommt den Erwartungswert, der sich aus den Unterknoten errechnet, zugewiesen. Bei Knoten, bei denen der Spieler selbst entscheiden kann, was er spielt, wird natürlich weiterhin der Weg mit dem maximalen Unterknoten gewählt.

Abb. 5 veranschaulicht dieses Prinzip an einem einfachen Beispiel:

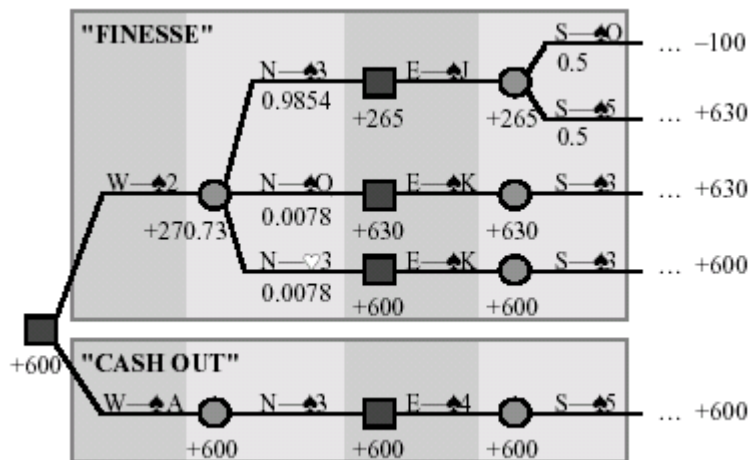


Abbildung 5. Auswertung eines möglichen Baumes. Die Endknoten besitzen eine Wertung. Den Vorängerknoten wird jeweils der Erwartungswert zugewiesen. Das Ergebnis ist entscheidend dafür, welche Strategie verwendet wird.

3.4 Ergebnisse

Eine naheliegende Frage, die sich nun stellt, ist natürlich ob der soeben beschriebene Ansatz tatsächlich auch in der Praxis effizient ist. Zwar werden die Berechnungen dadurch deutlich einfacher, was jedoch noch lange nicht bedeutet, dass auch bessere Ergebnisse geliefert werden. Dennoch fällt bei Tignum 2 auch die Bewertung unter diesem Aspekt positiv aus.

Versuche, in denen eine erweiterte Version des Bridge Baron mit Tignum 2 gegen die Vorgängerversion spielten, lieferten als Ergebnis, dass die Weiterentwicklung als deutlich besser eingestuft werden kann.

Auch bei einem Turnier der besten kommerziell erhältlichen Bridge-Programme schnitt der Bridge Baron am besten ab (Abb. 6 - Quelle: [AIM98]):

Programm	Land	Ergebnis
Bridge Baron	USA	1. Platz
Q-Plus	Deutschland	2. Platz
Microbridge 8	Japan	3. Platz
Meadowlark	USA	4. Platz
GIB	USA	5. Platz

Abbildung 6. Ergebnisse der Baron Barclay World Bridge Computer Challenge.

4 Weitere Ansätze

Natürlich ist *HTN*-Planung nicht die einzige Technik, die bei Bridge-Programmen angewendet wird, wenn auch die derzeit effektivste. Alternativ wurden bereits auch zwei andere Ansätze erprobt: Die vollständige Suche und die Monte Carlo Technik.

4.1 Vollständige Suche

Der Computer stellt, ausgehend vom vorangehenden Reizen Vermutungen auf, wo sich welche Karten befinden. Anschließend durchläuft das Programm im Spiel alle möglichen Knoten dieser vermuteten Verteilung und entscheidet davon ausgehend welche Karte gespielt wird. Die Problematik dieser Variante ist, dass sich die Kartenverteilung durch das Reizen nur ungefähr abschätzen lässt und somit auch oft falsche Vermutungen aufgestellt werden. Auch lässt sich das Programm offensichtlich leicht durch gezieltes Reizen täuschen. Alpha Bridge ist ein Programm, das diese Technik implementiert. Die Tatsache, dass dieses Programm 1992 den letzten Platz bei einer Computer Olympiade belegt hat, zeigt aber, dass sich dieser Ansatz nicht als sonderlich effizient herausgestellt hat.

4.2 Monte Carlo Technik

Ein anderer Ansatz ist die sogenannte Monte Carlo Technik. Dabei generiert der Computer eine größere Anzahl von zufälligen Kartenverteilungen und trifft seine Entscheidungen nach dem durchschnittlichen Ergebnis, das ein bestimmter Zug bei allen ermittelten Kartenverteilungen bewirken würde. Dadurch kann nach dem Gesetz der großen Zahlen davon ausgegangen werden, dass im Mittel die jeweils beste Entscheidung getroffen wird. Leider bleibt auch bei diesem Vorgehen noch eine verhältnismäßig große Anzahl an Knoten, die

untersucht werden müssen. Bei einer weiteren Vereinfachung, die deswegen gemacht werden musste, handelt es sich deshalb um das Zusammenfassen mehrerer als äquivalent einstuftbarer Karten. So macht es für das eigentliche Spiel oft keinen Unterschied, ob zum Beispiel eine 3 oder eine 4 einer Farbe gespielt wird. Die beiden Karten lassen sich deshalb auch bei diesem Ansatz zu einem Fall zusammenfassen. Analog findet man noch viele weitere Situationen, die in der Praxis keinen nennenswerten Unterschied ausmachen. Hierdurch kann man die Knotenzahl also ebenfalls deutlich verringern, auch wenn sich nicht so viele Karten als äquivalent einstuft lassen, wie es beim Betrachten der Strategien der Fall ist. Dadurch lässt sich der Baum soweit reduzieren, dass letztendlich doch komplette Durchläufe mit akzeptablem Zeitaufwand möglich sind. Dennoch hat sich auch diese Methode nicht als überaus effizient erwiesen, weil sie einen nicht zu umgehenden Nachteil mit sich bringt: Das Programm geht davon aus, dass es die Verteilung der Karten kennt. Dadurch können Informationen, die man zum Beispiel durchs Reizen oder durch ein Erkundungsspiel erhält nicht verwertet werden. Diese Informationen sind beim Bridge-Spiel jedoch durchaus wertvoll und offensichtlich nicht zu vernachlässigen.

5 Zusammenfassung

Zusammenfassend lässt sich sagen, dass *HTN* momentan wohl einen verhältnismäßig vielversprechenden Ansatz in Bezug auf künstliche Intelligenz beim Bridge darstellt. Zwar findet es dort erst seit vergleichsweise kurzer Zeit Anwendung, allerdings lassen die ersten Ergebnisse darauf schließen, dass noch weit mehr Potential in diesem Ansatz steckt. Auch in anderen Gebieten, wie zum Beispiel die bereits erwähnte Herstellung von Mikrowellen-Übertragungs-Modulen wird *HTN* bereits mit Erfolg eingesetzt. Dennoch ist die Verwendung in Computer-Spielen, bei denen der komplette Zustand des Spiels bekannt ist, eher unwahrscheinlich, da sich dort die Brute-Force-Ansätze in Verbindung mit weiten Optimierungen als deutlich effektiver erwiesen haben. Anders hingegen verhält es sich bei Spielen wie Bridge, bei denen jedem Spieler nur ein Teil des Gesamtzustands bekannt ist. Die Effizienz von *HTN*-Planung in Verbindung mit Bridge kommt von der Tatsache, dass Bridge ein Spiel ist, bei dem die Planung und die Anwendung diverser Strategien eine entscheidende Rolle spielen. Bridge Baron geht mit seiner Implementierung von *HTN* damit so an das Spiel heran, wie es auch ein menschlicher Spieler am ehesten machen würde.

Es bleibt natürlich die Frage, welcher der Ansätze sich bei Bridge oder vergleichbaren Spielen langfristig als der beste erweisen wird. Mit Sicherheit lässt sich darüber noch keine Aussage machen. Konventionelle Brute-Force-

Implementierungen werden dafür vermutlich nicht in Frage kommen, da die immense Anzahl zu prüfender Knoten auch in absehbarer Zeit noch nicht von aktuellen Computern zu bewältigen sein wird. Auf der anderen Seite hat der hier beschriebene Ansatz den Nachteil, auf eine sehr begrenzte Anzahl an Fällen eingeschränkt zu sein. Zwar steht die Anwendung der taktischen Elemente bei Bridge im Vordergrund, dennoch kann Bridge deswegen lange nicht ausschließlich auf diese reduziert werden. Dadurch werden bei diesem Ansatz immer eine Menge an Fällen bleiben, die vom Computer nicht berücksichtigt werden, einem menschlichen Spieler aber sehr wohl ins Auge fallen könnten.

Literatur

- [Bri03] Bridge Lernprogramm, Deutscher Bridge-Verband e.V. <http://www.bridgeverband.de/>
- [AIM98] Stephen J. J. Smith, Dana S. Nau, Tom A. Throop. Computer Bridge: A Big Win for AI Planning. *AI Magazine*, 19(2):93-105, June 1998
- [AAAI98] Stephen J. J. Smith, Dana S. Nau, Tom A. Throop. Success in Spades: Using AI Planning Techniques to Win the World Championship of Computer Bridge. *IAAI-98/AAAI-98 Proceedings*, pp. 1079-1086
- [Com96] Stephen J. J. Smith, Dana S. Nau, Tom A. Throop. A Planning Approach To Declarer Play In Contract Bridge. *Computational Intelligence*, Volume 12, Number 1, 1996
- [OP91] K. Currie and A. Tate. O-Plan: The open planning architecture. *Artificial Intelligence*, 52(1):46-86, 1991
- [SIPE99] D. E. Wilkins. Using the SIPE-2 Planning System: A Manual for SIPE-2, Version 6.1. SRI International Artificial Intelligence Center, 333 Ravenswood Ave., Menlo Park, California 94025, July 1999.
- [UMCP] K. Erol, J. Hendler, D. S. Nau. UMCP: A sound and complete procedure for hierarchical task network planning. In Hammond [Ham94], pages 88-96.
- [Ham94] K. Hammond, editor. *Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems (AIPS-94)*, University of Chicago, Illinois, USA, June 13-15 1994. AAAI Press, Menlo Park, USA.