

# **Fakultät für Informatik, Universität Ulm**

**Sommersemester 06**

## **Proseminar künstliche Intelligenz**

**Thema:**

Bridge Baron: Karten spielen gegen den Computer

**Markus Hipp**

`markus.hipp@informatik.uni-ulm.de`

**Zusammenfassung** In dieser Arbeit wird auf den Einfluss von künstlicher Intelligenz bei der Kartenspielsoftware Bridge Baron eingegangen. Die Software versucht, Spiele anhand menschlicher Strategien durchzuführen, also sich dem menschlichen Verhalten anzupassen. In Spielen wie Schach werden stattdessen schlicht alle möglichen Züge des Spielers bzw. seines Gegners berechnet. Da es sich bei Bridge aber um ein Kartenspiel und somit um ein so genanntes 'imperfect information game' handelt, würde die Berechnung aller Möglichkeiten den zeitlichen Rahmen sprengen. Unter einem 'imperfect information game' versteht man ein Spiel, in dem der Spieler nicht den vollständigen Zustand seiner Umwelt kennt, wie zum Beispiel die Verteilung der Karten unter den anderen Mitspielern.

## 1 Einleitung

Seit ihrer Erfindung haben Computer eine unglaubliche Entwicklung erfahren. Rechnerleistungen übersteigen heutzutage leicht die des menschlichen Gehirns. So berechnete der IBM Rechner "deep blue" im Duell mit Schachweltmeister Kasparov bereits 1997 ungefähr 60 Milliarden Möglichkeiten pro Zug, während ein menschlicher Spieler höchstens ein paar Dutzend Spielzüge simuliert ehe er sich für einen entscheidet.

Seit langem sucht man den Vergleich Computer gegen Mensch, welcher sich am einfachsten in Form von Spielesoftware austragen lässt. Doch in gewissen Spielen, z.B. bei Bridge sind Rechner trotz ihrer enormen Leistung nicht in der Lage die besten Spieler in lokalen Bridge Clubs zu schlagen. Hier setzt nun die künstliche Intelligenz an. Denn anders als Schach oder Dame ist Bridge ein so genanntes 'imperfect information game', was bedeutet, dass nicht der komplette Zustand der Umwelt bekannt ist. Denn die Verteilung der Karten auf die Spieler ist zufällig. Allein die eigenen Karten sind bekannt. Diese Unbekannte lässt aufgrund der enormen Anzahl eine Simulation aller möglichen Spielzüge nicht zu. Um nun den Rechenaufwand so gering wie möglich zu halten, wird mit der so genannten HTN-Planungsmethode versucht, das Verhalten der Software dem des menschlichen Spielers anzugleichen. Die Software soll lernen, nach Strategien zu agieren. Da es wesentlich weniger Möglichkeiten für verschiedene Strategien gibt als Möglichkeiten für auszuspielende Karten, wird versucht, die Anzahl der möglichen Ergebnisse so klein zu halten, dass diese anschließend komplett durchlaufen werden können. Im Folgenden wird nun die Vorgehensweise dieser HTN Planung erklärt und mit herkömmlichen Methoden, die bei anderer Software verwendet werden, verglichen. Außerdem wird ein konkretes Beispiel in Bridge angeführt, welches die Abläufe deutlich macht. Um ein Besseres Verständnis von Bridge zu erlangen, werden die Grundlagen dieses Kartenspiels im folgenden Kapitel kurz erklärt.

## **2 Das Kartenspiel Bridge kurz vorgestellt [Bri03]**

### **2.1 Grundlagen**

Bridge wird mit vier Spielern gespielt, welche sich an einem viereckigen Tisch gegenüber sitzen. Gespielt wird mit einem normalen Kartendeck, bestehend aus 52 Karten. Diese sind in die vier Farben Pik, Herz, Kreuz und Karo aufgeteilt. Joker werden nicht verwendet. Ausgeteilt wird gleichermaßen an alle Spieler am Tisch. Jeder Spieler beginnt also mit 13 Karten.

Um die einzelnen Spieler besser identifizieren zu können, benutzt man oft auch die vier Himmelsrichtungen die ihre Sitzposition beschreiben. Den Spieler, der unten sitzt, bezeichnet man mit Süd u.s.w.

Das eigentliche Spiel setzt sich aus zwei Grundbestandteilen zusammen: Dem Reizen und dem eigentlichen Spiel mit den Karten.

Bridge ist ein Partnerschaftsspiel, das heißt, die Spieler die sich am Tisch gegenüber sitzen, spielen als Team gegen die beiden anderen Spieler. Grundsätzlich ist das Ziel bei Bridge so viele Stiche wie möglich mit dem jeweiligen Partner zu machen.

Beim Reizen versucht jeder Spieler seine Kartenstärke einzuschätzen und anzusagen, wie viele der 13 möglichen Stiche er gemeinsam mit seinem Partner, dessen Karten er zu diesem Zeitpunkt noch nicht kennt, macht. Die Spieler reizen im Uhrzeigersinn, solange bis kein Spieler das letzte Gebot mehr überbieten kann.

Der Spieler, der das höchste Gebot abgegeben hat wird zum Alleinspieler. Der Spieler links vom Alleinspieler muss die erste Karte ausspielen. Der Partner des Alleinspielers wird ab sofort zum "Dummy", denn er muss seine Karten offen und für jeden sichtbar auf den Tisch legen und darf nicht mehr am Spielgeschehen teilnehmen. Ab jetzt übernimmt der Alleinspieler das Spiel des Dummys. Ziel des Alleinspielers ist nun, mit seinen Karten und den Karten des Dummys, die angesagte Anzahl von Stichen zu machen oder zu überbieten.

Um einen Stich zu machen, spielen alle Spieler nacheinander im Uhrzeigersinn eine ihrer Karten aus. Alle Spieler müssen die Farbe der Karte, die als erstes ausgespielt wird bekennen, wenn z.B. die erste Karte Pik ist, müssen die folgenden Spieler auf jeden Fall Pik spielen, sofern sie Pik besitzen.

Eine Farbe kann beim Reizen vom Alleinspieler als Trumpf bestimmt werden, dies ist aber nicht zwangsläufig notwendig. Alle Karten dieser angesagten Farbe sind nun Trumpfkarten.

Trümpfe stechen alle Karten anderer Farbe. Ein Spiel ohne Trumpf ist für den Alleinspieler auch möglich. Wenn ein Spieler eine Farbe in einem Stich nicht bekennen kann, so darf er eine Karte seiner Wahl spielen. Diesen Vorgang nennt man abwerfen, wenn der Spieler eine Karte spielt, die nicht Trumpf ist, denn der Stich kann auf diese Weise auf keinen Fall gewonnen werden. Die andere Möglichkeit ist, den Stich mit einem Trumpf zu stechen, falls man einen solchen besitzt. Wenn alle vier Spieler ihre Karten gespielt haben gewinnt die höchste Karte den Stich.

Der Spieler der letztendlich den Stich macht, ist an der Reihe die nächste Karte auszuspielen. Dieser Vorgang wird solange wiederholt, bis alle Karten gespielt worden sind.

Wenn der Alleinspieler am Ende des Spiels die angesagte Anzahl an Stichen erreicht, gewinnt er mit seinem Partner, dem Dummy das Spiel, wenn nicht, verlieren Beide und ihre Gegner gewinnen.

## **2.2 Verschiedene Strategien**

Bridge ist ein relativ leicht zu erlernendes Spiel, doch um es gut spielen zu können, bedarf es einiger Übung und Geduld.

Das zentrale taktische Element stellt in Bridge der Dummy dar, bzw. die Tatsache, dass Dieser seine Karten offen und für alle einsehbar auf den Tisch legen muss. Dies ist ein klarer Vorteil für den Alleinspieler, da die Stärke des Blattes sofort ersichtlich ist. Daraus kann der Alleinspieler natürlich viel schneller und effektiver sein Spiel entwickeln und bestimmte Taktiken auf sein Blatt anwenden.

Doch auch die Spieler der Gegenpartei können durch diese Tatsache Vorteile aus ihrem Spiel ziehen. Genauso können sie, wie auch der Alleinspieler, diverse Taktiken anwenden, die alle dasselbe Ziel verfolgen: So viele Stiche wie möglich zu erzielen.

Einzelne Taktiken, die später noch im Zusammenhang mit der künstlichen Intelligenz auftauchen, werden dort noch genauer erläutert.

### **3 Brute force search vs. HTN**

#### **3.1 Brute force bei bekannter Spielesoftware**

Häufig wird bei Spielesoftware (meist bei Brettspielen) die Brute force search Methode benutzt, welche sich bei vielen Spielen bewährt hat. Vor allem bei Spielen, wie Schach, Dame oder Othello ist die Software häufig besser als fast alle menschlichen Spieler, wie auch der Schachcomputer „deep blue“ vor Jahren schon eindrucksvoll bewiesen hat. Noch deutlicher fällt der Unterschied bei Software für Dame und Othello aus. Bei beiden Spielen gibt es wohl keinen menschlichen Spieler, der den Computer schlagen könnte.

#### **3.2 Vorgehensweise von Brute force search (erschöpfende Suche)**

Die Brute force Methode berechnet einfach in jedem Schritt des Spiels, jede Möglichkeit, wie der nächste Spielzug aussehen könnte, und berechnet daraufhin jede Möglichkeit wie der Gegner auf diesen Zug reagieren könnte um dann wieder die eigenen Möglichkeiten zu berechnen und so weiter, bis zum Schluss alle möglichen Spielverläufe erforscht sind.

Alle diese Möglichkeiten werden in einen Suchbaum eingetragen und einfach komplett durchsucht nach dem Pfad, der für den Spieler den größten Erfolg verspricht. Die Wurzel dieses Baums ist der Spielstart und jede weitere Ebene beschreibt einen Zug im Spiel bis zu den Baumblättern, die das Ende des Spiels darstellen.

Jede Ebene stellt einen Zug eines Spielers dar, d.h. aus der Sicht eines Spielers ist jede zweite Ebene nicht von ihm beeinflussbar, da sein Gegner am Zug ist. Diese Tatsache schränkt den Spieler in gewisser Weise ein. Hier kommt die so genannte MinMax- Formel zum Einsatz:

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

Abbildung 1. minmax Formel

Nachdem der Suchbaum komplett aufgestellt worden ist, wird mit Hilfe die minmax- Formel der beste Pfad für den Spieler gesucht. Der beste Pfad wird durch den Erfolg des Spielers bestimmt, den er hat wenn er einen Zug macht. Wenn er selber am Zug ist, wird er den Zug machen mit dem er später den bestmöglichen Erfolg erzielen kann ( $\max\{\text{minimax}(q) : q \text{ is a child of } p\}$  if it is our move at the node  $p$ ). Andererseits wird er, wenn sein Gegner am Zug ist, für sich den kleinstmöglichen Erfolg einplanen, da er ja davon ausgehen muss, dass sein Gegner auch gewinnen will ( $\min\{\text{minimax}(q) : q \text{ is a child of } p\}$  if it is our opponent's move at the node  $p$ ). Diese Vorgehensweise kann gut am Beispiel Tic Tac Toe demonstriert werden.

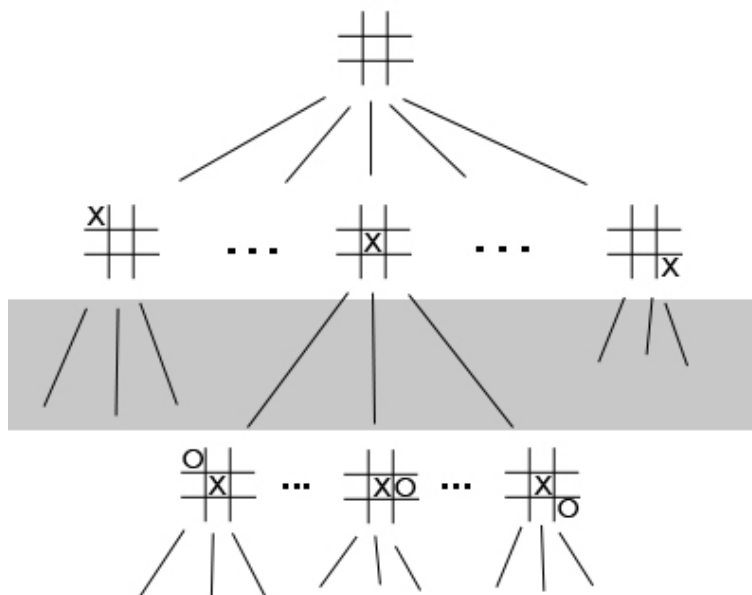


Abbildung 2. Brute Force am Beispiel Tic Tac Toe

Wurzel und somit der Spielbeginn ist das leere Spielfeld. Der Spieler hat nun im ersten Zug (angenommen er ist x) 9 verschiedene Möglichkeiten sein Kreuz zu machen. Im darauf folgenden Schritt kann der Gegner an den 8 verbleibenden Kästchen sein o setzen (grauer Bereich in Abb. 2). Nach diesem Prinzip wird der gesamte Baum entwickelt.

Zum Schluss bleiben dann als Blätter die komplett ausgefüllten Spielfelder stehen. Aus diesen Spielfeldern, können schon diejenigen Pfade gestrichen werden, die nicht zu einem Sieg führen. Die Züge, die zu diesem Ergebnis führen können getrost vernachlässigt werden. Auch schon Züge in oberen Knoten können gestrichen werden, wenn alle Nachfolgeaktionen nicht zum gewünschten Sieg führen. Natürlich können nur Entscheidungen des Spielers weggelassen werden. Bei den Entscheidungen des Gegners müssen alle Möglichkeiten betrachtet werden. Auf diese Weise fallen nicht nur einzelne Pfade weg, sondern ganze Teilbäume unter dem gestrichenen Knoten. Dadurch kann man die Anzahl der gewinnbringenden Pfade minimieren. Jedoch muss in jedem Fall erstmal der komplette Baum aufgestellt werden, denn die Formel arbeitet sich von den Blättern hoch bis zur Wurzel.

Die Brute Force Methode ist, wie schon der Name sagt (erschöpfende Suche) und auch das obige Beispiel belegt, auf jeden Fall mit immensem Rechenaufwand verbunden. So hat der Baum im Tic Tac Toe Beispiel alleine  $9!$  Blätter ( $9! = 362880$ ). Der Rechenaufwand für ein ähnliches Schach Beispiel wäre noch deutlich höher. Diese Vorgehensweise lohnt sich also nur, wenn der Baum eine gewisse Größe nicht überschreitet bzw. wenn genügend Zeit zur Verfügung steht um die Berechnungen auszuführen. Dann jedoch ist sie sehr effektiv, da wirklich jede Möglichkeit in Betracht gezogen wird.

### 3.3 HTN Ein Überblick

HTN (Hierarchical Task Network) ist ein Planungsverfahren mit dem abstrakte Pläne konkretisiert werden können. So wird ein Plan in kleinere Pläne oder Methoden aufgeteilt, soweit bis nur noch elementare Handlungen übrig bleiben, die leicht auszuführen sind. Diese Art von Problemlösung wird auch Dekomposition genannt. Ein Problem wird hierbei nach und nach in Teilprobleme geteilt bis diese leicht zu lösen sind. Mit dieser Methode lassen sich anfangs schwierige Probleme ziemlich genau und einfach lösen. Doch nicht nur Probleme können auf diese Weise gelöst werden, sondern auch komplexere Abläufe. Ein Beispiel:

Man will von A nach B reisen. Es existiert die Methode  $travel(A,B)$ . Nun versucht man dieses Vorhaben mit größtmöglichem Erfolg für den Benutzer zu lösen, denn es gibt verschiedene Möglichkeiten von A nach B zu kommen.

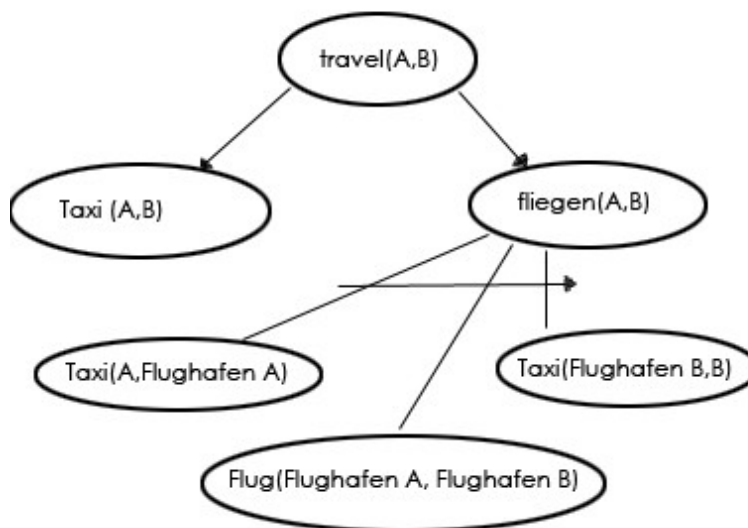


Abbildung3. Dekompositionshierarchie

In Betracht gezogen werden hier nun die zwei Möglichkeiten, entweder zu fliegen oder mit dem Taxi zu fahren. Beide Methoden werden auf ihre Erfüllbarkeit überprüft und dann die bessere von beiden gewählt. Methoden haben gewisse Übergabeparameter und können Abbruchbedingungen besitzen. z.B. könnte es nicht möglich sein Taxi zu fahren ohne Geld zu Besitzen.

Wenn man sich für die Flugzeugvariante entscheidet, muss man automatisch den Plan mit drei untergeordneten Aufgaben (subtasks) erfüllen. Dazu gehören die Methoden

*Taxi(A,FlughafenA)*, *Flug(FlughafenA,FlughafenB)*, *Taxi(FlughafenB,B)*. Diese müssen in genau dieser Reihenfolge abgearbeitet werden. Dieser Plan existiert bereits, d.h. er kann einfach abgerufen und eingesetzt werden, und muss nicht jedes Mal Stück für Stück neu geplant werden.

HTN entwickelt also nur Pfade am Baum, die für den Benutzer relevant sind und nicht blind alle Möglichen. So werden im obigen Beispiel Möglichkeiten wie 'Zu Fuß gehen' oder 'Fahrradfahren' gar nicht beachtet, obwohl sie theoretisch möglich gewesen wären, aber für den Benutzer keinerlei positive Wirkung gehabt hätten.

### **3.4 Brute force search vs. HTN**

Beide Methoden wurden in den letzten Punkten etwas näher vorgestellt. Nun ist es an der Zeit, beide Methoden miteinander zu vergleichen um Vor- bzw. Nachteile in ihren Vorgehensweisen herauszufinden und zu überprüfen welche Methode bei welcher Art von Spiel besser arbeitet.

Einen sehr großen Vorteil hat die Brute Force Methode in Spielen, bei denen der Spieler den aktuellen Zustand der Welt in jedem Zug der des Spiels genau kennt. In diese Kategorie fallen genau die schon oben genannten Spiele Schach, Dame, Tic Tac Toe. Bei diesen Spielen kann per Brute Force Search relativ leicht ein Ergebnisbaum konstruiert werden.

Anders bei so genannten 'imperfect information games'. So werden Spiele bezeichnet, bei denen der Spieler eben nicht den kompletten Zustand der Welt kennt. Das Paradebeispiel hierfür sind alle Arten von Kartenspielen unter anderem auch Bridge. Bei dieser Art von Spiel müssen nicht nur die Möglichkeiten betrachtet werden, welche Karte der Gegner spielt, sondern auch die Verteilung der Karten unter den verschiedenen Spielern. Dem Spieler sind nur seine eigenen Karten bekannt und evtl. Karten die im Laufe des Spiels schon gespielt worden sind. Jedoch ist die Verteilung der Restkarten auf die verbleibenden Spieler noch dem Zufall überlassen. Aus diesem Grund würde die Größe eines möglichen Ergebnisbaums jeden Rahmen sprengen. Denn es müssten bei dessen Berechnung auch alle möglichen Kartenverteilungen betrachtet werden.

In diesem Punkt kann HTN deutlich bessere Ergebnisse liefern. Ein Vergleich der beiden Methoden angewandt auf Bridge liefert folgendes Ergebnis:

	Brute Force	HTN
worst case	ca. $5,6 \times 10^{44}$ Knoten	ca. 305000 Knoten
average case	ca. $2,3 \times 10^{24}$ Knoten	ca. 26000 Knoten

Im schlechtesten Fall hätte der Ergebnisbaum des Brute force search bei Bridge ca.  $5,6 \times 10^{44}$  Knoten. Der Baum, der mit HTN entwickelt worden wäre hätte hingegen nur 305000 Knoten. Ähnlich deutlich fällt das Ergebnis im Durchschnittsfall aus.

Wie nun HTN auf einen so kleinen Ergebnisbaum kommt wird nun im folgenden Punkt erklärt.

## 4 HTN in Bridge

### 4.1 Vorgehensweise von Tignum2

Tignum2 ist die Implementierung von HTN in der Software Bridge Baron. Grundlage für die Anwendung von HTN bei Bridge ist die Tatsache, dass Bridge wie schon anfangs erklärt, sehr viele taktische Elemente vereint. Auf diesen taktischen Ansätzen baut Tignum2 auf, indem es nicht versucht, alle Möglichkeiten die es für den Spielverlauf gibt aufzustellen, sondern den Lösungsbaum nach Strategien zu entwickeln, die für das Blatt des Spielers angemessen scheinen. Auch auf Grund der Tatsache, dass eine Bridge Partie nicht länger als 10 Minuten dauern sollte, ist eine Brute force search unmöglich zu realisieren. Anstatt also alle Möglichkeiten für den Spieler zu betrachten, eine beliebige Karte zu spielen, werden nur ein Paar Strategien (je nach Blatt des Spielers verschieden viele) in Betracht gezogen und abhängig von Ihnen nur eine kleine Anzahl von spielbaren Karten.

So wird verhindert, dass Karten, die für eine Strategie nicht in Frage kommen, überhaupt simuliert werden. Somit fällt nicht nur die Berechnung zum Spielen dieser einen Karte weg, sondern auch die Berechnung des kompletten Teilbaums, die auf das Spielen dieser Karte folgen würde. Die Software simuliert ein Verhalten des menschlichen Spielers, welcher auch nur Karten spielt, welche für ihn gewinnbringend zu sein scheinen, je nach Strategie, die er verfolgt. Bei allen anderen Karten würde er auch nicht versuchen, die Folgen ihres Ausspielens zu vermuten, weil er sie von Anfang an gar nicht spielen würde. So spart der Mensch, wie auch ein Rechner den größten Teil seiner Energie für die Simulation der

Spielzüge, die überhaupt nur in Frage kommen. Diese Ergebnisbäume, die nur einen Bruchteil der kompletten Ergebnisbäume darstellen können dann mit Brute force search komplett durchlaufen werden.

Zwar kennt Tignum2 nicht immer den kompletten Zustand seiner Umwelt, jedoch ist ein gewisser Teil der Umwelt immer bekannt. Für die Berechnung dieses Zustandes benutzt Tignum2 zwei verschiedene Art von Methoden: Mit so genannten state information sets wird der Status aller Karten beschrieben, deren Position sicher bekannt ist. Dazu gehören die eigenen Karten und Karten, die schon gespielt worden sind. Für alle anderen Karten und Spielsituationen gibt es belief functions, welche die Position der Karten mit verschiedenen Wahrscheinlichkeiten bei den verschiedenen Mitspielern vermuten. Diese Wahrscheinlichkeiten werden nach jedem Spielzug aktualisiert. So wird nach und nach der Zustand der Umwelt für den Spieler immer bekannter.

Mit Hilfe dieser Methoden werden die Pfade des Lösungsbaums mit Wahrscheinlichkeiten belegt und mit deren Hilfe der Pfad verfolgt, der mit der größten Wahrscheinlichkeit zum Erfolg führt.

#### **4.2 Erfolge von HTN Planung bei Bridge**

Anders als bei Brettspielen, ist Software bei Kartenspielen oft nicht in der Lage mittelmäßige menschliche Spieler zu schlagen. Auch in Bridge ist die Software oft schon den Spielern lokaler Bridge Clubs unterlegen. Jedoch setzte die Implementierung von Tignum2 in Bridge für Software neue Maßstäbe.

Die Berechnung der Lösungsbäume wird dadurch zwar immer einfacher und schneller, jedoch ist dies ja noch keine Garantie für den Erfolg. Letztendlich setzte sich Tignum2 in der erstmaligen Implementierung bei Bridge Baron 8 1997 jedoch bei der Baron Barclay World Bridge Challenge gegen vier weitere Konkurrenten aus der ganzen Welt durch. Die Baron Barclay World Bridge Challenge ist ein Wettbewerb zwischen Bridge Computerprogrammen. Bridge Baron 8 wurde zum stärksten Bridgeprogramm auf dem Markt.

### 4.3 Funktionsweise von HTN an konkretem Beispiel

Um genauer auf die Vorgehensweise von HTN bei Bridge einzugehen, betrachten wir nun eine ganz konkrete Spielsituation in Bridge aus Sicht des Alleinspielers, der im Beispiel im Osten sitzt.

Für das Verständnis der nächsten beiden Abbildungen bzw. ihrer Erklärungen werden nun kurz zwei hier angewandte Taktiken erklärt:

Schnitt (engl. Finesse):

Der Partner (Nord) besitzt mehrere hohe, jedoch nicht aufeinander folgende Karten (z.B. Ass und Dame). Die Karte, die sich dazwischen befindet (König) ist im Besitz des Gegners (West). Wenn nun der Spieler (Süd) den Stich mit einer niedrigen Karte dieser Farbe eröffnet, so können unter Umständen beide Karten des Partners stechen, obwohl der König des Gegners die Dame des Partners theoretisch stechen würde. Denn wenn der Gegner (West) mit dem König sticht kann der Partner sein Ass spielen und ab diesem Zeitpunkt wäre seine Dame die höchste Karte im Spiel. Somit hätte der Partner zwei Stiche (abgesehen von Trumpfstichen) sicher. Andererseits kann der Partner mit der Dame stechen, falls der Gegner (West) nicht den König spielt. Auch damit würde seine Dame stechen und er hätte sein Ass aber auch noch auf der Hand um den König des Gegners zu stechen.

Hochspielen (engl. Cash out):

Eine der einfachsten Taktiken in Bridge. Man spielt die höchste Karte einer Farbe im Spiel und macht den Stich sicher. Diese Taktik ist vor allem bei Spielen ohne Trumpf sehr effektiv. Hohe Karten können ohne Trumpf nicht gestochen werden. Besitzt man also zum Beispiel Ass, König und Dame einer Farbe, so sind das drei sichere Stiche (ohne Trumpf) und man kann alle nacheinander ohne Bedenken ausspielen.

Bei Spielen mit Trumpf ist Hochspielen nur bedingt erfolgreich, da vielleicht nicht jeder die angespielte Farbe bedienen kann und somit die Chance hat, den Stich mit einem Trumpf zu machen.

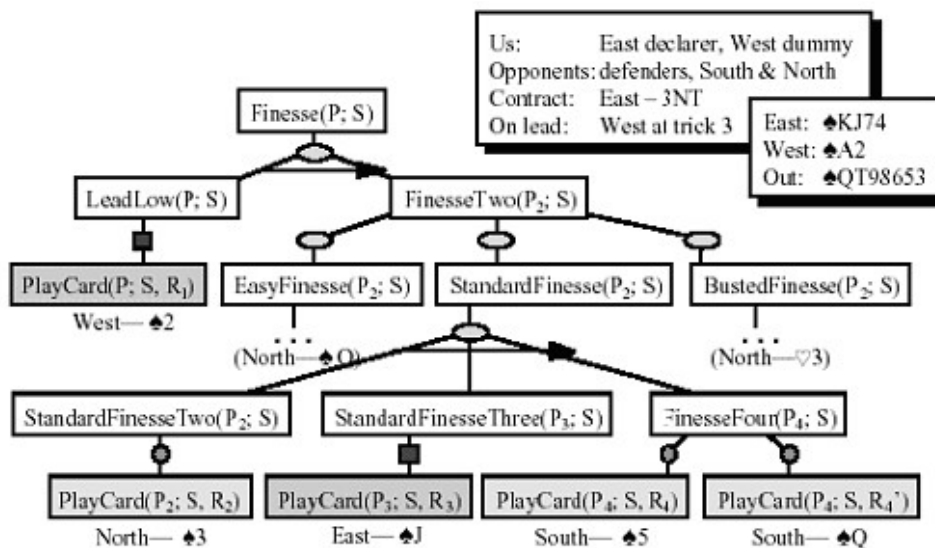


Abbildung4. Planungsbaum für Finesse Strategie

Abbildung 4 zeigt eine Planungshierarchie an einem konkreten Beispiel bei Bridge. Die konkreten Pläne sind hier die Taktiken welche in Bridge verwendet werden. Die hier gespielte Taktik ist die Finesse Taktik.

Den Stich eröffnet der Alleinspieler mit den Karten des Dummys. Der Alleinspieler könnte einen Schnitt versuchen, da die Voraussetzungen dafür gegeben sind. Der Dummy ist im Besitz des Asses und der Zwei. Der Alleinspieler hat den König und den Buben und kann damit auf jeden Fall den Stich machen, falls der Dummy mit der Zwei eröffnet. Die *Schnitt* Taktik (Finesse) wird natürlich nur dann überhaupt vom Programm in Betracht gezogen, wenn die Voraussetzungen dafür gegeben sind. Diese Abfrage wird in entsprechenden Methoden überprüft. Falls nun, wie in unserem Fall, die *Finesse(P;S)* Methode aufgerufen wird und alle Voraussetzungen gegeben sind, gelangen wir im Planungsbaum eine Ebene tiefer. Hier muss der Plan, bestehend aus den beiden Methoden *LeadLow()* und *FinesseTwo()*, komplett und in richtiger Reihenfolge durchlaufen werden. Innerhalb der *FinesseTwo()* Methode, gibt es nun verschiedene Möglichkeiten für den Gegner auf die ausgespielte Karte in der *LeadLow()* Methode zu reagieren. Dieser Schritt ist natürlich vom Alleinspieler nicht beeinflussbar, da die Karten des Gegners ihm nicht bekannt sind. Der Baum, wird nach jeder Aktion des Gegners neu erstellt, da sich die Zustände der Welt ständig ändern. Jedoch sind in unserem Beispiel nicht mehr viele Möglichkeiten übrig, denn der Alleinspieler kennt seine eigenen Karten bzw. die des Dummys und weiß welche Karten schon gespielt worden sind. Allein die Dame des Gegners würde am Spielverlauf etwas ändern. Die dritte Möglichkeit wäre, dass der Gegner kein Pik mehr auf der Hand hat.

Die Umsetzung des Planbaums zu einem Ergebnisbaum hätte folgendes Ergebnis:

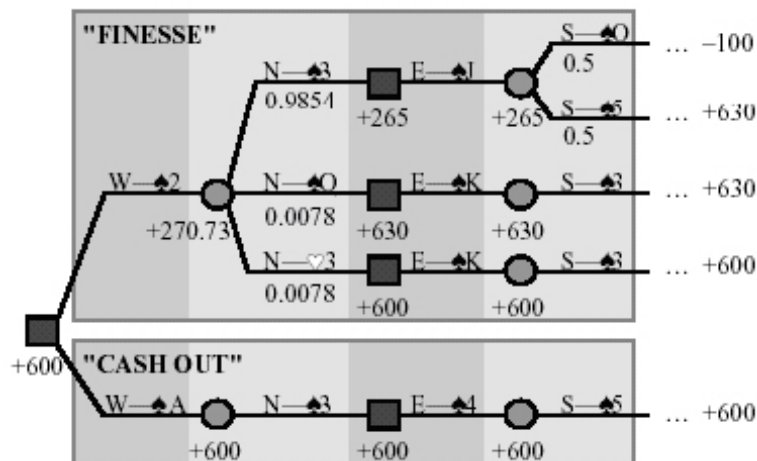


Abbildung5. Ergebnisbaum

Außer des Schnittes wären im Beispiel auch noch andere Strategien denkbar. Alle Strategien werden nun gemeinsam in einem Baum betrachtet. Außerdem werden Wahrscheinlichkeiten und verschiedene Erwartungswerte hinzugefügt. Züge des Gegners werden mit Wahrscheinlichkeiten belegt, während eigene Züge mit Erwartungswerten abhängig von den Wahrscheinlichkeiten des Teilbaumes belegt werden, der auf den eigenen Zug folgt. Wie gut zu erkennen ist, hat der Baum eine überschaubare Größe, zumindest was den ersten Stich anbelangt. Abbildung5 zeigt jeden Zug der einzelnen Spieler. West beginnt, gefolgt von Nord, dann Ost und zum Schluss Süd. Wie bei der Cash out Taktik zu erkennen ist, wird hier nicht unterschieden welches Pik von Nord gespielt wird, da der Stich auf jeden Fall an West geht. Der Dummy hat nur die Möglichkeit zwischen Pik Ass und Pik Zwei. Nur diese Zwei Karten stehen zur Auswahl gespielt zu werden, eine andere Karte zu spielen kommt in dieser Situation nicht in Frage und muss deshalb gar nicht erst simuliert werden.

## **5 Zusammenfassung**

Nicht nur bei Bridge hat sich im Softwarebereich HTN Planung durchgesetzt, sondern auch in anderen Implementierungen kommt HTN immer mehr zur Verwendung. Die HTN Planung hat auf jeden Fall das Potential sich weiter durchzusetzen, zumal der Ansatz noch relativ jung ist und noch Zeit braucht sich entscheidend weiter zu entwickeln. Im Bezug auf Spielesoftware hat HTN wohl nur Chancen im Bereich der 'imperfect information games', also bei Spielen in denen nicht der komplette Zustand der Umwelt bekannt ist. Bei diesen Spielen setzt sich der Ansatz durch, Handlungen nach Strategien zu planen und durchzuführen, anstatt stur alle Möglichkeiten zu berechnen. Hier orientiert man sich am Verhalten an menschlichen Spielern. Von einem Angleichen an dieses Verhalten kann jedoch nicht die Rede sein. Denn so gut man das Planen von Strategie auch umsetzen kann, fehlt etwas wie z.B. ein Bauchgefühl, das das Spiel eines menschlichen Spielers durchaus prägen kann. Außerdem wird bei diesem Ansatz immer davon ausgegangen, der Gegner hat ebenfalls Ahnung vom Spiel und verhält sich dem entsprechend. Nur was, wenn der Gegner das Spiel nicht so gut beherrscht, wie das die Software annimmt, oder gar mit Absicht der Strategie entsprechend falsche Spielzüge ausführt? Es wird wohl noch eine Zeit lang dauern, bis eine Bridgesoftware mit menschlichen Spielern mithalten kann indem sie nicht nur versteift auf Strategien seine Züge macht, sondern den Überblick über das Spiel bekommt.

## 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