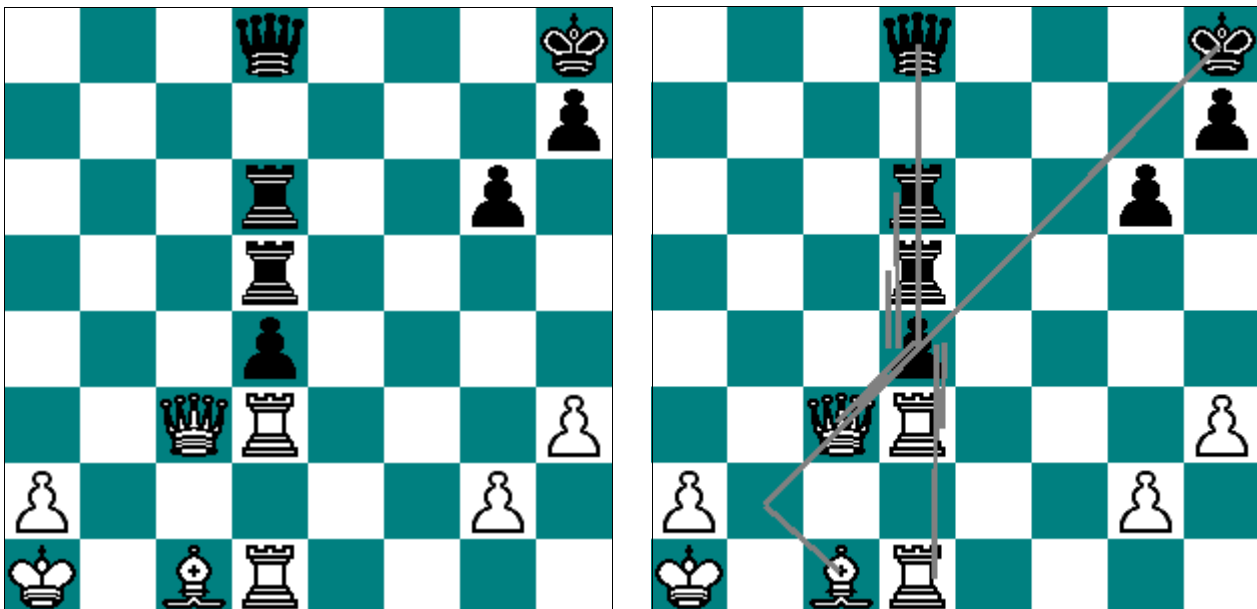


Computerschach

Der Unterschied zwischen Mensch und Maschine

Zur Wahrnehmung und Bewertung von Schachszenen

Um den Unterschied zwischen Schachcomputer, Schachprofi und Schachamateur aufzuzeigen, führte Alfred Binet 1894 erste Experimente durch, deren Ergebnisse er in seiner Veröffentlichung *Psychologie des grands calculateurs et joueurs d'échecs* darlegte. Binets Annahme vor Beginn der Untersuchung war, daß erfahrene Spieler quasi photographische Abbilder der Schachbretter in ihrem Geist aufbauen. Seine Untersuchungen zeigten, daß Schach drei Kategorien von mentalen Fähigkeiten erfordert: a) erworbenes Wissen aus Studienmaterial und früher gespielten Partien, b) Gedächtnis, um dieses Wissen und die bisherigen Züge zu speichern und c) abstraktes Vorstellungsvormögen, um den nächsten Zug zu finden, und die daraus resultierende Position zu ermitteln.

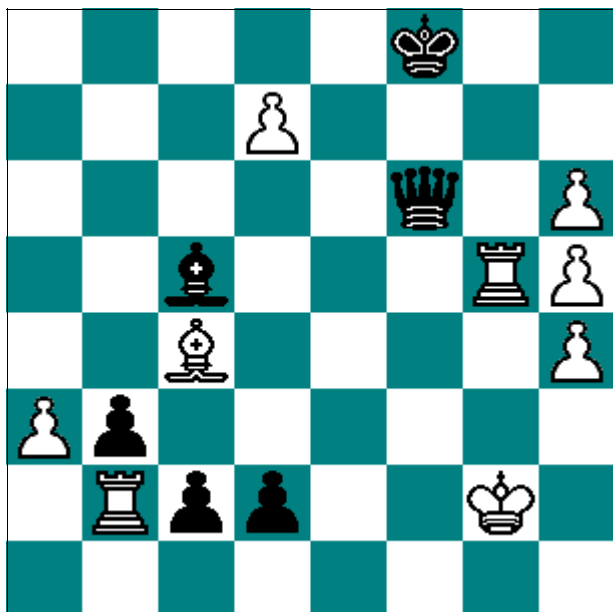


Einer von Binets Probanden rekonstruierte damals seine Überlegungen für die obige Schachszene, die in der linken Zeichnung dargestellt wurden. Interessant dabei sind die für die derzeitige Situation unwichtigen und daher unberücksichtigten Bauern auf dem Königsflügel, und die starke Betonung der für die unmittelbare Spielentwicklung wichtigen Linien.

1927 wurden von Djakow, Petrowski und Rudik in der damaligen UdSSR erstmalig gut konzipierte Versuche durchgeführt. Sie führten auch Experimente zur Wahrnehmung durch, in dem sie erfahrenen Schachspielern ein Brett mit bis zu 25 Figuren für 5-10 Sekunden zeigten, und danach ihre Probanden aufforderten, die Situation nachzustellen. Es ergab sich eine direkte Proportionalität zwischen Schachkenntnissen und korrekt positionierten Figuren, wobei Schachprofis bis zu 95% Trefferquote erzielten.

Intuitives Erfassen vs. Berechnen

Von physiologischen Aspekten abgesehen zeigt sich hier, daß Schachkünstler ein Abbild der Spielsituation besitzen, deren Vor- und Nachteile sie teilweise intuitiv erfassen. Hierbei spielen Begriffe wie Pattern Matching, und die logische Struktur der Situation eine Rolle, wobei der Zusammenhang zwischen der logischen Spielsituation und der Fehlerhäufigkeit ebenfalls bekannt ist und mit Hilfe der nächsten Spielsituation nachgewiesen werden kann, die willkürliches Positionieren der Figuren erzeugt wurde. Diese Situation dürfte kaum in einer normalen Spielsituation erreicht werden, direkt proportional zu ihrer Unwahrscheinlichkeit ist die Fehlerquote beim Nachstellen des kurz gezeigten Spiel-



brettes.

Für dieses geistige Abbild einer Situation existiert in der moderneren Psychologie der Begriff des Mentalen Modells, welches Weltwissen, Situationswissen - derzeitige Position aller Figuren - und Regelwissen - erlaubte Züge jeder Figur, Matt- und Patt-Zustand - zu einem Abbild der aktuellen Situation verknüpft. Mentale Modelle sind derzeit Forschungsgegenstand in mehreren Untersuchungen¹.

Für einen Schachcomputer unterscheiden sich die beiden abgebildeten Situationen hingegen nicht: beide Situationen werden programmintern repräsentiert und durchgerechnet. Eine intuitive Erkenntnis von starken oder schwachen Linien, von wichtigen oder gefährdeten Feldern fehlt (bisher).

Zur Berechnung und Bewertung einer Spielsituation von einem Schachcomputer existiert eine Vielzahl

von Algorithmen, auf die im folgenden Abschnitt eingegangen wird. Allerdings erweitern moderne Computerschachprogramme die recht starren Algorithmen um weitere Komponenten wie Pattern Matching (für typische Spielsituationen und Figurstrukturen), Zeitmanagement (optimales Nutzen der vorhandenen Zeit für das Spiel, nicht je Zug) oder Datenbanken (Eröffnungs-, Endspiel- und Matt-Datenbanken) - dazu später mehr.

Berechnungsfunktionen

Ein Zug setzt sich aus zwei Halbzügen (je einen für jede Farbe) zusammen, Halbzüge werden auch als ply bezeichnet. Eine mögliche Zugfolge kann bevorzugt als Teilbaum repräsentiert werden, dessen Höhe mit der Anzahl der vorausgerechneten Züge korrespondiert. Für die Berechnung einer Zugfolge innerhalb dieser Baumstruktur existieren eine Vielzahl von Algorithmen, hier eine Auswahl:

Brute Force

Als Brute Force wird Berechnung jedes möglichen - bzgl. der Spielsituation noch so unsinnigen - plies bezeichnet. Der Rechen- und Speicheraufwand steigt hierbei exponentiell an, womit die vorausgerechnete Zugfolge gering bleibt, allerdings wird keine Kombination übersehen.

Brute Force-Algorithmen wurden speziell im Anfang der Computerschachprogramme genutzt, also vor 1960.

MiniMax und NegaMax

Bei MiniMax und NegaMax wird der Suchbaum mit gewichteten Knoten dargestellt. Die Gewichtung entspricht dabei einer Spielstandsbewertung. Um einen möglichen besten Zug zu finden, muß also das Maximum in einer Tiefe gesucht werden. Da beide Spieler abwechselnd am Zug sind und aus ihrer jeweiligen Position den besten Zug ermitteln wollen, ergibt sich ein Wechsel zwischen Minimum und Maximum.

Um die Verwechslung zwischen den beiden Spielern zu verhindern, wird die Knotengewichtung immer vom Standpunkt des Spielers, der am Zug ist, vorgenommen, d.h. durch Negierung der Gewichtung vom Standpunkt des Gegenspielers. Diese Methode wird als NegaMax bezeichnet, der sich in Pseudo-C-Code² wie folgt darstellt:

¹ Als Beispiel mögen die beiden Sätze Hans war auf dem Weg zur Schule. Er machte sich Sorgen wegen der Mathematikstunde. Er hatte Angst, er würde die Klasse nicht unter Kontrolle halten können. und Der Lehrer war auf dem Weg zur Schule. Er macht sich Sorgen wegen der Mathematikstunde. Er hatte Angst, er würde die Klasse nicht unter Kontrolle halten können. dienen. Ein Großteil der Leser benötigt zum Lesen des ersten Satzes signifikant mehr Zeit, da das Mentale Modell zunächst von einem Schüler namens Hans ausgeht, dieses Modell aber revidieren muß, als klar wird, daß das gewählte Szenario falsch ist. Im zweiten Satz hingegen ist sofort klar, daß es sich um den Lehrer dieser Klasse handelt, die Wahl des Szenarios ist von Anfang an eindeutig.
(Versuche von Sanford und Garrod, 1981 und 1983)

```

int NegaMax(pos, depth) {
    if (depth == 0)
        return(Evaluate(pos));
    best = -INFINITY;
    succ = Successors(pos);
    while (not Empty(succ)) {
        pos = RemoveOne(succ);
        value = -NegaMax(pos, depth-1);
        if (value > best)
            best = value;
    }
    return(best);
}

```

Die Anzahl der zu bearbeitenden Positionen beträgt B^H mit

B : Breite des Baumes (durchschnittlich mögliche Zugzahl)
H : Höhe des Baumes.

$\alpha\beta$ -Search

Dieser Algorithmus versucht, die Anzahl der Positionen in der NegaMax- bzw. MinMax-Suche zu verringern und dadurch größere Suchtiefen zu erreichen. Hierbei wird davon ausgegangen, daß große Teile des Suchbaumes nicht mit ihren exakten Zahlen bekannt sein müssen, eine Aussage, ob diese besser oder schlechter als bisherige Ergebnisse sind, reicht aus. Zu diesem Zweck werden zwei weitere Parameter übergeben:

```

int AlphaBeta(pos, depth, alpha, beta) {
    if (depth == 0)
        return Evaluate(pos);
    best = -INFINITY;
    succ = Successors(pos);
    while (not Empty(succ) AND best < beta) {
        pos = RemoveOne(succ);
        if (best > alpha)
            alpha = best;
        value = -AlphaBeta(pos, depth-1, -beta, -alpha);
        if (value > best)
            best = value;
    }
    return(best);
}

```

Der große Vorteil hier liegt in dem früheren Abbruch der while-Schleife, wenn der Wert der Variable best den von beta erreicht oder übersteigt. Dieser Fall wird als *Cutoff* bezeichnet und ist auf das Spiel bezogen eine korrekte Entscheidung, denn dies bedeutet, daß dieser Teilbaum schlechter als die übergebene Version ist.

Im Optimalfall muß dieser Algorithmus $B^{(D+1)/2} + W^{D+2} - 1$ Positionen berechnen

Aspiration Search

stellt eine geringe Verbesserung der $\alpha\beta$ -Suche dar: anstelle des dortigen Startaufrufes AlphaBeta(pos, depth, -INFINITY, +INFINITY) werden einzelne Teilbäume durch eine ϵ - Umge-

2 Die Variablen und Funktionen haben dabei folgende Bedeutung:

pos : Eine Position in einem Schachspiel
depth : Die Tiefe des Suchbaumes
Evaluate() : Eine Bewertungsfunktion für die aktuelle Spielsituation aus Sicht des Spielers am Zug. Wird bei einer realen Implementation anhand des Materialwertes und einiger Positionsvorschriften berechnet. Liefert einen Wert zwischen -INFINITY und +INFINITY zurück
best : Das Maximum der Bewertungsfunktion während der Untersuchung der nächsten Ebene
Successors() : Eine Funktion zur Ermittlung aller möglichen Halbzüge für den aktuellen Spielstand
succ : Eine Menge von möglichen, zu bewertenden Halbzügen.

bung um das erwartete Ergebnis eliminiert. Der Aufruf ändert sich somit zu AlphaBeta(pos, depth, value-window, value+window). Durch diese vorgegebene Limits werden zwar weniger Teilbäume durchsucht, allerdings kann das Ergebnis aus der ϵ -Umgebung herausfallen und eine erneute Suche erzwingen.

Principal Variation Search

Stellt ebenfalls eine Variation der $\alpha\beta$ -Suche dar, die zugrundeliegende Idee geht von der Annahme aus, daß bei einer perfekten Zugfolge und einem Variationsfenster von $\beta = \alpha + 1$ alle Zugfolgen außerhalb dieses Fensters eine schlechtere Bewertung erhalten, als diese perfekte Zugfolge selbst. Dabei wird weiterhin die Entwicklung des Suchfensters beobachtet, anhand dessen Entwicklung entweder eine neue $\alpha\beta$ -Suche mit vollständigem Suchfenster gestartet oder diese Zugfolge gewählt.

```
int PrincipalVariation(pos, depth, alpha, beta) {
    if (depth == 0)
        return Evaluate(pos);
    succ = Successors(pos);
    pos = RemoveOne(succ);
    best = -PrincipalVariation(pos, depth-1, -beta, -alpha);
    while (NOT Empty(succ) AND best < beta) {
        pos = RemoveOne(succ);
        if (best > alpha)
            alpha = best;
        value = -PrincipalVariation(pos, depth-1, -alpha-1, alpha);
        if (value > alpha AND value < beta)
            best = -PrincipalVariation(pos, depth-1, -beta, -value);
        else if (value > best)
            best = value;
    }
    return best;
}
```

Eine weitere Verbesserung dieses Algorithmus ist auch als NegaScout bekannt.

Memory Enhanced Test

Stellen eine Familie von Suchalgorithmen dar, die zunächst eine $\alpha\beta$ -Suche mit einem minimal Fenster $\beta = \alpha + 1$ (ein Bauernwert Unterschied) starten, dann aber verschiedene Vorgehensweisen bei der Zugauswahl und damit in der Sequenz der Bewertungen implementieren. Meistens wird hierbei der Aufruf der Suchfunktion sehr oft hintereinander mit verschiedenen Parametern vorgenommen und die bereits berechneten Position in einer (großen) Hash-Tabelle gehalten (siehe nächsten Punkt: Transposition table).

Transposition table

Stellt eine Hash-Tabelle über bereits berechnete Positionen dar, die versucht, einmal bewertete Positionen erneut zu verwenden. Eine derartige Hash-Tabelle kann bis zu 75% des benötigten Speichers ersparen, erzeugt allerdings die Gefahr eines Remis durch Zugwiederholung, da zwei Positionen nicht dieselbe Zugvorgeschichte besitzen.

Iterative Deepening

Dieser Algorithmus ruft wiederholt schnell hintereinander eine Suchroutine mit begrenzter Suchtiefe auf und erhöht diese Suchtiefe sukzessive, bis ein Rechenzeitlimit überschritten oder die maximale Suchtiefe erreicht wurde. Der Vorteil dieser Methode ist zu nennen, daß das jeweils letzte Ergebnis bereits zur Verfügung steht und nicht zunächst eine maximale Suchtiefe vorgegeben werden muß. Dieser Algorithmus wird meist zusammen mit Hash-Tabellen verwendet oder als Eingabe für die erweiterten $\alpha\beta$ -Algorithmen.

Killer heuristic

Dieser Ansatz soll Zug-Reihenfolgen optimieren; er geht dabei von der Annahme aus, daß ein guter Zug in einem Teilbaum ebenfalls in einem anderen Teilbaum gleicher Tiefe einen guten Zug darstellt. Daher berechnet dieser Algorithmus für jeden Halbzug ein bis zwei Killer-Züge vor restlichen Zügen.

History heuristic

Stellt ebenfalls eine Methode zur Optimierung der Zugreihenfolge dar, hierbei wird eine Tabelle mit Von → Nach - Bereichsstatistiken bzgl. der Spielfelder geführt und bei der Zugoptimierung verwendet.

Null move heuristic

versucht die Suche zu verkürzen, in dem in bestimmten Situationen - bei einer bereits guten Bewertung der Position - ein Nullzug gemacht, d.h. auf einen Zug verzichtet wird und mit verringerter Tiefe gesucht wird. Wenn das Ergebnis dieser Suche besser als Beta ist, wird auf eine weitere Suche verzichtet, ansonsten wird die normale Suche fortgesetzt.

Dieser Test birgt die große Gefahr des Übersehens tieferer Spielkombinationen in sich, kann aber sehr große Teilbäume des Spieles eliminieren und so einen Zeitgewinn bedeuten.

Andere Bewertungskomponenten

Neben der allgemein angewandten rekursiven Suchfunktion auf Zugbäumen spielen in den meisten Computerschachprogrammen noch weitere Bewertungskomponenten hinein, die bekanntesten sind z.B.:

Materialbewertung und -vorteile

Materialbewertung erscheint zunächst einfach: ein Bauer erhält einen Wert von 1, Turm einen Wert von 5, Läufer und Springer einen Wert von 3, Dame 9 und der König einen unendlichen Wert - sein Verlust bedeutet schließlich das Ende vom Spiel (obige Werte entstammen der Repräsentation von Deep Blue). Ältere Schachcomputer bewerten die Figuren noch anders, hier zählt der Turm 4 Einheiten.

Eine Materialbewertung geschieht nun durch simples Addieren der vorhandenen Figurwerte. Allerdings sind hier einige Sonderfälle zu berücksichtigen, ein Bauer auf der 7. oder 2. Linie, der kurz vor der Umwandlung in eine höherwertige Figur steht, muß entsprechend höher bewertet werden, eine Kalkulation, bei der viele ältere Schachprogramme versagen.

Positionsgewichtung

Frühere Schachprogramme und Spieler gingen von dem Grundsatz aus, das die Beherrschung des Brettzentrums spielentscheidend ist - von hier aus sind alle anderen Positionen ideal erreichbar, eine dort plazierte starke Figur besitzt hier ihren größtmöglichen Einfluß. Aus diesem Grund beginnen viele der herkömmlichen Eröffnungen mit Zügen wie e2-e4 oder d2-d4.

Einige Großmeister haben inzwischen Eröffnungen entwickelt, die die Entwicklung des Zentrums in der Absicht verzögern, daß der Gegner seine Struktur zu sehr öffnet und Lücken in seiner Verteidigung entstehen.

In diesem Zusammenhang ist auch die ideale Eröffnung a2-a4 zu sehen, die ein Schachprogramm Mitte der 70er-Jahre fand. Nach diesen Analysen sei eine solche Eröffnung ein Garant für den Gewinn.

Situation des Königs

Ein Schachprogramm ordnet idealerweise seinem eigenen König immer einen möglichst hohen Verteidigungswert zu. Dieser kann durch Material- und Positionsgewichtung errechnet werden. Im Verlauf der Partie ändert sich diese Bewertung von möglichst defensiv (Eröffnung und Mittelspiel) zu offensiv (Endspiel mit wenigen Figuren, wenn auch der König eine aktive Rolle übernimmt).

Tempogewinn

Ein Tempogewinn stellt einen spielfördernden, entwickelnden Zug dar, während der Gegner diesen Zug mit einem Nullzug, der seine Situation nicht verbessert, beantwortet. Damit hat der Spieler mit dem Tempogewinn gegenüber seinen Gegner erzielt und seine Spielsituation verbessert.

Andere Programmkomponenten

Eröffnungs- und Endspiel-Datenbanken

Die guten Anfangszüge des Schachspiels sind teilweise bis zum 20. Zug in ihren möglichen Varianten analysiert und bewertet worden. Bis in die 80er Jahre hinein gab es Schachbücher zu kaufen,

in den nur entsprechende Varianten abgedruckt wurde - speziell eine Reihe aus der DDR, geschrieben von mehreren russischen Großmeistern, die systematisch alle Varianten einer Eröffnung, teilweise über mehrere Bände verteilt, analysierten.

Auch das Endspiel mit bestimmten Konstellationen (z.B. Turm und König gegen König) sind analysiert und deren Gewinnchancen ermittelt worden, so daß hier teilweise systematisch nach einem festen Algorithmus vorgegangen werden kann.

Ende der 80er-Jahre, als die Mephisto-Schachcomputer der Firma Hegener und Glaser beinahe acht Computerweltmeisterschaften hintereinander gewannen - einer der namhaften Programmierer war z.B. Richard Lang - gab es für ihre Computer Erweiterungsmodule, die die eingebauten Eröffnungsbibliotheken um weitere Variationen - etwa 80.000 Züge - ergänzten. Inzwischen gibt es diese Schachcomputer kaum noch, sie wurden von Schachprogrammen wie Fritz 3, Fritz 4 oder Chess 5000, die herkömmliche Wintel-Systeme nutzen, verdrängt. Aber auch für diese Programme gibt es zusätzliche Erweiterungen in einem normierten Datenformat auf CD-ROM zu kaufen, teilweise enthalten diese bis zu 400.000 Züge.

Matt-Datenbanken

Größere Schachcomputer wie Deep Blue (32 IBM RS/6000-Knoten mit je 8 VLSI-Prozessoren, das Betriebssystem AIX und das IBM Parallel Messaging System MPI nutzend) oder Intel Paragon Parallel Supercomputer (1824 Prozessoren mit je 16 bzw. 32 MB RAM) nutzen im Endspiel einen Teil ihrer Rechenkapazität, um von eingespeicherten Mattsituationen quasi rückwärts rechnend durch Hinzufügen von Spielfiguren und Spielzügen zur derzeitigen Situation zu gelangen. Hierzu werden im System - speziell bei Deep Blue ehemalige Partien Kasparovs - möglichst viele Schachpartien abgelegt.

Starke vs. Schwache KI

Zu der Begriffswelt der Künstlichen Intelligenz gehören die Begriffe *Starke KI* und *Schwache KI*, die verschiedene Lösungsansätze und deren Qualität definieren: hiernach stellt *Schwache KI* die Lösung eines Problems mit Hilfe reiner Rechenleistung (um Kasparov zu zitieren: Quantität wird zu Qualität) und eines begrenzten Regelwissens durch systematische Iteration über alle Möglichkeiten, deren Bewertung und Sortierung dar. *Starke KI* hingegen bedeutet die Lösung eines Problems mit Hilfe von KI-Konzepten wie Pattern Matching, Schlußfolgerung und Wissen. Die KI-Forschung begrenzt sich inzwischen selbst auf die Probleme der Starken KI und erklärt Probleme der Schwachen KI als gelöst - es fehlt meistens nur an genügend Rechenkapazität.

Alle bisher vorgestellten Bewertungsfunktionen und deren Ansatz der rekursiven Bewertung von Schachsituationen sind eindeutig der Schwachen KI zuzuordnen, weshalb Schachprogramme während der letzten Jahre kaum noch Forschungsgegenstand der KI-Forschung waren.

Um so interessanter ist die gesellschaftliche Diskussion des Sieges von Deep Blue über Kasparov, die Reaktionen der Zuschauer auf diesen Sieg und nachträgliche Besprechung der Partien.

Geschichtliches

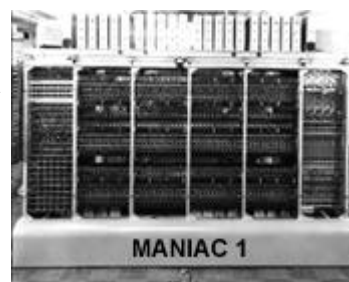
Die Anfänge

- 1894 Alfred Binet führt erste Experimente zur Wahrnehmung und Verarbeitung von Schachpositionen durch. Er entdeckt drei mentale Fähigkeiten, die einen guten Schachspieler auszeichnen: erworbenes Wissen, Gedächtnis und abstrakte Vorstellungskraft.
- 1927 Djakow, Pietrowski und Rudik führen mit acht russischen Großmeistern Untersuchungen zur Wahrnehmung und Rekonstruktion kurz gezeigter Schachpositionen durch.
- 1947 Alan Turing entwickelt das erste Schachprogramm, das korrekte Züge berechnen und eine Position bewerten konnte. Mangels



eines existierenden Computers simuliert er dessen Berechnungen selbst.

- 1949 Der Computer Ferranti löst Matt-in-2-Aufgaben
- 1956 MANIAC I (0.011 MIPS, entwickelt in Los Alamos) spielt auf einem 6x6-Brett Schach. Dies ist das erste funktionsfähige Schachprogramm
- 1957 Bernstein entwickelt ein Schachprogramm für eine IBM 704. Das erste full-fledged Schachspiel eines Computers



Das Zeitalter der Expertensysteme

- 1966 Auf einem PDP-6 läuft MacHack VI, entwickelt am MIT von Greenblatt, erreicht bei dem Massachussets Amateur Wettbewerb 1 Unentschieden und 4 verlorene Partien, USCF rating 1243.
- 1968 David Levy wettet einen Betrag von 3.000 US-\$, daß er 10 Jahre lang von keinem Computerschachprogramm besiegt wird.
- 1974 Das russische Schachprogramm KAISSA, auf einem englischen Rechner entwickelt, gewinnt die Welt-Computerschachmeisterschaften in Stockholm mit 4:0.
- 1977 Erste Mikroschachcomputer, CHESS CHALLENGER wurde entwickelt.
- 1977 Das Schachprogramm CHESS 4.5 erreichte eine USCF Bewertung von 2271.
- 1977 Michael Stean ist der erste Großmeister, der gegen einen Computer verliert (Blitzschach)



Das Zeitalter spezieller Chips

- 1982 wurde der Computer BELLE vom State Department der USA wegen Verstoßes gegen das Rüstungskontrollgesetz (Ausfuhrleistungsfähiger Systeme) beschlagnahmt, als es an einem Computerschachtourier teilnehmen sollte.
- 1983 BELLE erreicht eine Bewertung von 2263 (Schachmeister-Niveau)
- 1985 HITECH erreicht eine Bewertung von 2530
- 1988 DEEP THOUGHT erreicht eine Bewertung von 2745
- 1989 DEEP THOUGHT erreicht eine Bewertung von 2800, analysiert 2 Mio. Positionen/s. In einem Spiel gegen Kasparov verliert das Programm 0:2.



Das Zeitalter der Wintel-Systeme

- 1994 Kasparov verliert gegen FRITZ3 in München in einem Blitz-Turnier.
- 1995 Fritz3 gewinnt - einen handelsübliches Pentium90-System mit 16 MB RAM nutzend - gegen Star Socrates, der einen Intel Paragon Parallel Supercomputer nutze, mit 1824 Prozessoren à 16-32 MB
- 1997 Kasparov verliert 3,5 : 2,5 gegen DEEP BLUE

Der „perfekte“ Computer - HAL

Murray S. Campbell, Mitglied des Entwicklungsteams von Deep Blue - zusammen mit Feng-hsiung Hsu und A. Joseph Hoane Jr. - bei dem T.J. Watsons Research Lab von IBM und seit über 15 Jahren in der Computerschachforschung aktiv; schrieb einen interessanten Artikel über die Analogien und Gemeinsamkeiten des fiktiven Supercomputers HAL 9000 und Deep Blue. Die Eigenschaften von HAL 9000 resultieren aus der damaligen Sicht der KI-Forschung und den Erwartungen an Computerschachprogramme und sind daher ein interessanter Beitrag für eine Analyse historischer Sichtweisen.

HAL 9000 ist wesentlicher Bestandteil des Films 2001 - A Space Odyssey, der 1968 von Stanley Kubrick nach einer Kurzgeschichte von Arthur C. Clarke gedreht wurde. In diesem Film geht es um die 18-monatige Reise eines Astronautenteams zum Jupiter, der nach dem Fund eines außerirdischen Artefakts auf dem Mond stattfindet. An Bord schlafen die Besatzungsmitglieder, bis auf zwei Charaktere, in Hybernationskapseln, die beiden anderen steuern und kontrollieren zusammen mit dem Bordcomputer HAL 9000 den Flug des Schiffes. Während seiner Reise kommt es zu einer zunächst verdeckten Auseinandersetzung zwischen den beiden Astronauten und HAL. Dieser versucht, unter der Annahme, seine Intelligenz sei vollkommen und er könne keine Fehler machen, sich der menschlichen Besatzung zu entledigen, da er durch diese den Erfolg der Mission gefährdet sieht. Im Film kommt es schließlich zu einer Auseinandersetzung zwischen Frank Poole - als einzig überlebenden Astronauten - und HAL, der mit HALs Abschaltung endet.



Die Installierung von HAL als intelligente Einheit: Das Interview und die Schachsequenz

Um HAL als eine dem Menschen Frank Poole ebenbürtige und in einigen Bereichen überlegene Intelligenz einzuführen, verwendet Kubrick zwei Schlüsselszenen: zu einen läßt er HAL 9000 vor dem Start des Raumfahrzeuges ein Interview durchführen (etwa in der 56. Minute des Films), in diesem betont HAL, daß er foolproof and incapable of error sei. Dieser Satz stellt sich später als Schlüsselsatz für HALs Diskussionen mit Frank Poole dar.

Als zweite Schlüsselszene wird eine 30-sekündige Sequenz (etwa 60. Minute des Films) gezeigt, in der der Astronaut Frank Poole überlegen von HAL in einem Schachspiel besiegt wird. Hierzu verwendete Kubrick eine reale Partie aus dem Jahr 1913 in Hamburg, die ein Mattproblem in 3 Zügen beinhaltet und so einfach ist, daß der Kinozuschauer dieses Problem während des Films nachvollziehen kann. Gleichzeitig wurde das Schachspiel gewählt, da dies allgemein als komplexes Spiel bekannt ist. In dieser Szene ist Frank Poole mit Weiß am Zug und der folgende Dialog spielt sich in etwa ab:

Poole: Ummmm... anyway, Queen takes pawn
 HAL: Bishop takes Knight's Pawn
 Poole: Lovely move. Er.. Rook to King One.
 HAL: I'm sorry, Frank. I think you missed it. Queen to Bishop Three.
 Bishop takes Queen. Knight takes Bishop. Mate.
 Poole: Ah... Yeah, looks like you're right. I resign.
 HAL: Thank you for an enjoyable game.
 Poole: Yeah. Thank you.

Aussage und Bedeutung dieser Szene

Die Reaktion von Frank Poole wird von Campbell als sehr realistisch eingestuft: direkt nach der Demonstration der Zugfolge ist die Antwort- bzw. Reaktionszeit von Frank Poole kurz, aus dieser kurzen

Zeitspanne und seinem Tonfall - im englischen Originalton wesentlich besser zu erkennen - ist deutlich zu ersehen, daß er niemals annahm zu gewinnen und von der Korrektheit der Darstellung von HAL ausgeht. Obwohl Frank Poole als überdurchschnittlich intelligente Figur in diesem Film auftritt - dies ist wohl für die Leitung eines Raumschiffes notwendig -, wird deutlich, daß diese Sequenz nicht das Spiel zweier gleichstarker Gegner zeigt. Das Schachspiel dient Frank Poole nur zur Überwindung des 18-monatigen Fluges.

Im Film wird diese Haltung gegenüber HAL zu diesem Zeitpunkt vom Publikum akzeptiert, noch erscheint HAL mit weißer Weste. Dies steht allerdings im krassen Gegensatz zu der Realität, z.B. bei den Schachspielen Kasparov gegen Deep Blue, in denen Kasparov die menschliche Ehre gegen die Sizilienmonster verteidigen mußte und das Programmiererteam ausgebuht wurde.

Durch die Analyse des Gespräches zwischen HAL und Frank Pool während dieser Szene und dem Verlauf des Partie (bzw. durch deren Wahl durch den Regisseur) lassen sich einige Aussagen zu dem Schach-Stil von HAL treffen. Nach Campbell spielt HAL offensichtlich einen menschlichen Schachstil, im Gegensatz zu dem derzeitigen Computerprogrammstil, der sich auf massive Berechnungen (Brute Force) anstelle von intuitiven Erschließen stützt. Im Spiel selbst - Frank Pool mit den weißen Figuren, HAL mit den schwarzen, Spanische Eröffnung - verwendet HAL einen Zug³, dessen Auswahl mit Hilfe des MiniMax-Prinzipies nicht begründbar ist, sondern eine Falle darstellt. Diese Wahl erzeugt eine komplexe Situation, dessen naheliegender Zug einen Fehler darstellt. Mit dieser Wahl eines suboptimalen Zuges, der dennoch eine geschickte Falle für seinen menschlichen Gegner darstellt, setzt sich HAL von den derzeitig real existierenden Computerschachprogrammen ab, die solche Kombinationen (noch) nicht beherrschen.

Bei dieser Betrachtung sollte allerdings berücksichtigt werden, daß dieser Film aus der Perspektive des 60er Jahre entstand. Die Mehrheit der damaligen KI-Forscher waren wahrscheinlich der Annahme, daß Computer in der menschlichen Weise Schach spielen sollten, d.h. durch explicit reasoning über Zugfolgen unter der Zuhilfenahme intensiver Pattern-Matching-Tests. Diese Forderung hatte aber keinen durchschlagenden Erfolg, so daß in den 70er-Jahren andere Wege beschritten wurden.

HAL, der Turing-Test und Deep Blue

Wie im vorherigen Absatz dargestellt, wird HAL aus verschiedenen Gründen als ein Computer mit bzgl. des Schachspiels menschlichen Fähigkeiten dargestellt. Im krassen Gegensatz dazu steht Deep Blue, der als klassische Brute-Force-Maschine mit einigen Verbesserungen zur Erhöhung der selektiven Suchtiefe aufgebaut wurde.

Nach Turing besteht ein Computer seinen Test, wenn dessen Kommunikation in einem Blindversuch nicht mehr von einem Menschen unterscheidbar wäre. Diesen Test hätte HAL bzgl. des Schachspiels mit Leichtigkeit bestanden. Deep Blue hingegen besteht diesen Test (noch) nicht, allerdings gibt es einen interessanten Versuch von Frederic Friedel, dem Berater von Kasparov bei Computern, der Kasparov mehrere Partien von Deep Thought gegen menschliche Spieler vorlegte, und diesen aufforderte, den Computerspieler zu bestimmen. Kasparovs Analyse war in einigen Fällen falsch, er hielt Deep Thought für einen menschlichen Großmeister.

Deep Thought und Deep Blue fehlen noch einige Eigenschaften, die menschliche Spieler besitzen: hierzu gehört z.B. eine optimale Zeitnutzung, Konzeptverständnis (z.B. einer geschlossenen Abwehrkombination, einer Festung) und Reasoning, intuitives Erkennen bzw. Bewerten von Situationen. So ist von Kasparov eine Anekdote bekannt, in der er zu einer Spielsituation nach kurzer Bedenkzeit einen Gewinn für Weiß sah, aber nicht die genaue Zuganzahl und folge zu diesem Matt angeben konnte. Deep Blue hingegen konnte selbst nach längerer Analyse (mehrere Minuten) keine Gewinnchance erkennen.

Zusammenfassung

Computerschach hat im Laufe der letzten vier Jahrzehnte eine stürmische Entwicklung hinter sich: von

³ So behaupten die Analysten.

einfachen Automaten bis hin zum Sieg über den stärksten menschlichen Spieler. Die meisten derzeit existierenden Schachprogramme sind dem Großteil der Menschheit zwar überlegen, doch spielen sie (meist noch) einen anderen, erkennbare Schachstil.

Manche der Partien von Deep Blue können aber nicht mehr von menschlichen unterschieden werden, in diesem Fall besteht Deep Blue einen modifizierten Turing-Test. Diese Spiel-Intelligenz wird aber mit Methoden der Schwachen KI erreicht, d.h. durch sequentielles massives Berechnen variiertes Spielsituationen, weniger durch Ansätze der starken KI - Pattern Matching, Schlußfolgerung und Intuition. Einige Punkte werden allerdings von Deep Blue ansatzweise realisiert.

Nach dem Bestehen des Turing-Tests existiert für die Programmierer von Schachprogrammen keine große Herausforderung mehr, die von Deep Blue eingesetzte Rechenleistung wird in wenigen Jahren für wenig Geld erhältlich sein. Es bleibt die Verfeinerung bestehender Ansätze und der Schwenk zur Starken KI...

Literatur:

Herkömmlich:

F. Friedel: Meilensteine der Computerschach-Geschichte, 1990

F. Friedel: Nicht die Mutter aller Schachmaschinen, 1991

J. McCarthy: Chess as the Drosophila of AI, 1991

R. Levinson et al.: The Role of Chess in Artificial Intelligence Research

Online:

Chess Tree Search, Paul Verhelst

<http://www.xs4all.nl/~verhelst/chess/search.html>

Computer Chess Trivia, Bill Wall

<http://www.qnet.fi/mathiasen/cchess.html>

An Enjoyable Game - How HAL Plays Chess, Murray S. Campbell,

<http://mitpress.mit.edu/e-books/Hal/chap5/five1.html>

Die Internet Film Datenbank

<http://us.imdb.com/>

Deep Blue vs. Kasparov, IBM

<http://www.chess.ibm.com/>

What Has Research Shown About the Mental Differences

Between Chess Masters and Non Players?, Alleen Short,

<http://www.pomona.edu/Academics/Course/chess/papers/chessres.html>

The History of Modern Computers in General,

<http://bang.lanl.gov/video/sunedu/computer/comphist.html>

The PDP-6-Homepage

<http://www.ai.mit.edu/people/tk/pdp6/pdp6.htm>

Historic Computer Images

<http://ftp.arl.mil/ftp/historic-computers/>