Sächsisches Landesgymnasium Sankt Afra

BESONDERE LERNLEISTUNG

im Fachbereich Informatik

Einsatz von allgemeinen Evaluierungsheuristiken in Verbindung mit der Reinforcement-Learning-Strategie in der Schachprogrammierung

Der Versuch einer künstlichen Intelligenz

Albrecht Fiebiger

Interner Betreuer: Herr R. Böttcher
Externe Betreuer: Herr S. Meyer-Kahlen, Herr M. Block

Abgabedatum: 18.12.2008

Gliederung

Die externen Betreuer

Stefan Meyer-Kahlen

Stefan Meyer-Kahlen ist der Programmierer des Serienweltmeisters *Shredder*. Seit 1996 hat das Programm zwölf Titel als Schachcomputerweltmeister gewonnen und ist somit das erfolgreichste Programm aller Zeiten.

Vier der Titel waren Blitzweltmeisterschaften und einer wurde bei der Chess960-Weltmeisterschaft errungen. *Deep Shredder 11* hat eine Elozahl von 3050 in der SSDF-Weltrangliste vom 26. September 2008[1] und liegt damit auf dem dritten Platz.

Außerdem ist Meyer-Kahlen der Erfinder des Universal Chess Interface, des heutzutage populärsten Engineprotokolls, was eine Kommunikation zwischen Benutzeroberfläche und Programm realisiert.

Marco Block

Marco Block ist der Autor meiner Hauptquelle „Temporale-Differenz-Methoden im Schachmotor FUSc#". Er ist seit August 2004 Diplom-Informatiker[2] und beschäftigt sich als Mitarbeiter der Arbeitsgruppe „Künstliche Intelligenz" mit der Weiterentwicklung der E-Kreide und dem Computer-Vision-Projekt Saccadic, sowie leitet seit 2002 die „AG Schachprogrammierung" an der freien Universität Berlin.

[1] siehe: ssdf.bosjo.net/

[2] page.mi.fu-berlin.de/block/

1. Teil: Eine Einführung in die Schachprogrammierung

1.1 Einleitung

Wenn man in die Menschheitsgeschichte zurückblickt, gibt es viele Dinge, die sich stets wiederholten. Denn soviel sich die Menschen auch verändert haben, Menschen sind es trotzdem mit all ihren Eigenarten, Schwächen und Stärken geblieben. So findet sich schon zu jeder Zeit der uralte Traum, Wesen und Maschinen nach seinem Bilde zu erschaffen, die ihm dienen.

Lange Zeit hielt man dies für unmöglich, doch einsetzend mit den großen Neuerungen der Mechanik im 18. Jahrhundert erhielt auch diese Vision neue Nahrung und man hielt sich für fähig, nachdem es gelang eine Uhr zu konstruieren, das menschliche Denken nachzuahmen.

Schon damals galt Schach als ein „geeignetes Mittel für Experiment und Demonstration im unerforschten Reich des Denkens“[3] und so gab es unzählige Versuche, der Maschine das zweckmäßige Ziehen nahe zu bringen, aber am Ende stand stets die Erkenntnis, dass ohne Weiteres mit reiner Mechanik keine Intelligenz nachzubilden ist.

Dass es dadurch überhaupt keine schachspielenden Automaten gab ist dennoch nicht richtig, aber in ihrem Sensations- und Ruhmstreben wussten sich die jeweiligen Konstrukteure nicht anders zu helfen, als die perfekteste Nachahmung des Menschen zu schaffen: Den Menschen selbst, der unbemerkt im Inneren des neuen Weltwunders seinen Platz fand. Am berühmtesten ist wohl der schachspielende Türke, der von Baron von Kempelen erschaffen wurde. Er spielte gegen Napoleon I., Maria Theresia und viele weitere und schlug sie alle vernichtend. Obwohl tatsächlich ein Mensch in seinem Inneren verborgen war, handelt es sich trotzdem um eine Meisterleistung auf dem Gebiet der Konstruktion, da die Züge über ein äußerst kompliziertes Hebelsystem in das Innere der Maschine transportiert wurden und die Antwortzüge über ein ebensolches System von einer lebensgroßen Türkenfigur ausgeführt wurden. Weiterhin gilt der Türke als bahnbrechende Pioniertat in der Nachahmung menschlicher Akustik, da er mehr als 30 Wörter menschenähnlich sagen konnte.[4]

Aber damit, dass sich die Konstrukteure nur mit einer Täuschung zu helfen wussten, resignierten sie auch vor weiteren Versuchen, das menschliche Denken mit reiner

[3] Hoffmann/Hoffmann: “Schach unter der Lupe“, Sportverlag Berlin 1986, Seite 20

[4] www.schachcomputer.at/gesch1.htm

Mechanik nachzuahmen. Sie sind auch nicht zu beneiden; wie konnten sie das menschliche Denken nachahmen, wenn zu dieser Zeit noch nicht mal bekannt war, wie es funktioniert? Außerdem war natürlich das Schachwissen auch noch nicht sehr weit entwickelt, was heute meiner Meinung nach immer noch der Fall ist, auch wenn es riesig ist und sich unglaublich erweitert hat. Aber wie überall gilt auch im Schach: Alles ist relativ, es gibt nur eine Regel, die uneingeschränkt gilt und diese lautet: Die Partie ist zu Ende, wenn der König mattgesetzt wurde.

Alles Restliche ist davor Schall und Rauch und eine Stellungseinschätzung mit dem jetzigen Schachwissen dementsprechend unpräzise, da die Stellung noch so gut aussehen kann, wenn man trotzdem mattgesetzt wird. Und so lange man auf solche Ungenauigkeiten angewiesen ist, ist natürlich auch das Schachverständnis im Vergleich zur ‚Wahrheit' ungenau.

Demgemäß müssten die einzig ‚richtigen' Bewertungskriterien nur auf dem Matt fußen, welche aber nicht existieren, weil sich eine Stellung, in der wenigstens einzügig Matt gesetzt werden kann nicht eindeutig definieren lässt, dass alle Stellungen, die die Defintion erfüllen würden, tatsächlich ‚Mattstellungen' sind.

Daraus ergibt sich aber auch, dass der Mensch nie in der Lage sein wird, Schach perfekt zu spielen, denn er müsste tatsächlich alle möglichen Varianten bis zum Matt durchrechnen, da wie schon oben gesagt, das Matt das einzige gültige Kriterium einer Schachstellung sein kann.

Also kommt man zu der ernüchternden Feststellung, dass einzig und allein der Computer imstande sein wird, das Spiel perfekt zu spielen, wenn irgendwann einmal die entsprechende Hardware besteht. Doch wird dies zum Glück noch einige Zeit in Anspruch nehmen.

Meiner Ansicht nach lautet aber demgemäß die einzig und entscheidende Frage, wenn es darum geht, ob Computerschach gut oder schlecht ist: Wollen wir das Spiel perfektionieren? Und hier zeigt sich, der Leser möge mir verzeihen, eine gewisse Eitelkeit des Menschen. Er erzählt, wie reizvoll es für ihn ist, gerade nicht zu wissen, was die endgültige Wahrheit ist und nach ihr zu suchen beziehungsweise neue Wege auszuprobieren. Er findet die Faszination daran, dass sich jeder daran probieren, schöpferisch beteiligen und eigene „geniale" Ideen entwerfen kann. Würde dieser Reiz des Unerforschten abhanden kommen, würde das Schachspiel für viele auch an seiner Faszination verlieren und dies gilt natürlich umso mehr, wenn das Problem von einem Computer gelöst wurde.

Aber warum findet man Gefallen daran, die Wahrheit zu suchen, wenn man sie nicht finden will? Ist dies nicht ein Paradoxon in sich? Weswegen versucht man, ein Spiel zu beherrschen ohne wissen zu wollen, wie es zu beherrschen ist?
Weil Schach die Möglichkeit bietet, eigene Wege zu gehen und dadurch, dass es nicht die Wahrheit gibt, für jeden eigene Möglichkeiten bietet, kreativ zu sein. Aber damit ist diese Diskussion über die Lösung des Spiels natürlich nicht beendet, irgendwann wird sie existieren. Wir werden sehen, was dann passiert.
Ich hoffe aber, ein wenig von der Faszination des königlichen Spiels vermittelt zu haben und der Stellung des Computers innerhalb der Schachszene angerissen zu haben.
Die BeLL ist in drei Teile mit den unterschiedlichsten Aspekten über das Computerschach konzipiert, sodass hoffentlich für jeden etwas Interessantes zu finden ist. Für interessierte Laien werde ich im ersten Teil versuchen, das Computerschach näher zu bringen, im zweiten Teil wird es über meine Arbeit im Rahmen der BeLL gehen, die ich später noch erläutern werde. Beigefügt ist ein Sachregister, dass ich bei unverständlichen Begriffen zu konsultieren bitte. Im Buchband wird noch eine CD mit dem Quelltext sowie den kompilierten Versionen des Schachprogramms sowie des Reinforcement-Learning-Programms, welches noch erläutert wird, zu finden sein.
Weiterhin möchte ich versuchen, bei dem Einen oder Andern ein wenig Interesse für das Spiel zu wecken. Wenn dies geschieht, hat diese BeLL ihren Zweck erfüllt.

1.2 Stärken und Schwächen der Computer

Beginnen möchte ich mit einem kleinen Einblick darüber, was Computer zu leisten imstande sind und was sie nicht zu leisten vermögen.
Wie fast überall sind Stärken und Schwächen relativ, sodass man nur eine gute Beurteilung vornehmen kann, wenn man sich ein Bezugssystem festlegt. Ich werde hier, wie der Mensch im Allgemeinen, das Bezugssystem des Menschen nehmen und die Stärken und Schwächen des Schachcomputers anhand der menschlichen Fähigkeiten bewerten.
Ausgehend von den allgemeinen Eckdaten des Computerschachs ist es auch für den Laien ersichtlich, was die größten Unterschiede zwischen Mensch und Maschine ist: Die Maschine kann wesentlich schneller rechnen, der Mensch kann besser urteilen und planen. Der letzte Punkt ist vielleicht nicht ganz klar.
Schach ist natürlich viel zu komplex, um es einfach in Zahlen auszudrücken. Versuchen, eine perfekte Formel für Schach zu finden, würde bedeuten, von keiner Relativität der

Stellungsbewertungskriterien ausgehen, da man ihnen absolute Werte zuweisen muss. Aber alle der bekannten Bewertungskriterien haben Vor- und Nachteile, Ausnahmen und so weiter, die einfach unmöglich in einer Zahl auszudrücken sind. Ich sage absichtlich der bekannten, da ich nicht ausschließen will, dass es unbekannte gibt, die vielleicht doch noch einiges an „Wahrheit" liefern.

Ein weiteres prinzipielles Problem des Computerschachs ist, dass Sachen, die hinter dem Suchhorizont liegen, nicht erkannt und beurteilt werden können. Der Computer kann nur die von ihm errechnete resultierende Stellung bewerten, aber nicht, was sich daraus entwickeln kann. Dies ist ein weiterer großer Vorteil des Menschen: Obwohl er weit weniger tief rechnen kann, kann er die Endstellung auch auf ihre Möglichkeiten und Chancen für beide Seiten beurteilen und nicht nur statisch abzählen, was auf dem Brett steht.
Der Mensch ist fähig, in Plänen zu denken und demgemäß nicht jeden Zug einzeln zu berechnen, sondern Zugfolgen nach deren Sinn für den Gesamtplan gezielt auszuwählen. Man nennt dies selektives Vorgehen. Es ermöglicht dem Menschen, der geballten Rechenkraft des Programms etwas entgegenzusetzen, weil der Mensch viel besser die relevanten Zugfolgen erkennt und sie ebenso wie der Computer sinnvoll berechnen kann.
Aber diese Vorteile des Menschen werden immer geringer gegenüber dem Computer. Zwar gibt es immer noch Stellungen, die kein Computer in angemessener Zeit im Gegensatz zum Menschen richtig beurteilt, aber in einer praktischen Partie unter Zeitdruck ist es natürlich weit schwerer, dies zu realisieren, weil der Mensch einfach fehlbar ist und der Computer innerhalb seiner Möglichkeiten, die immer zunehmen, perfekt spielt.
Daraus resultiert auch ein weiterer gewaltiger Vorteil des Computers: Er ist gänzlich objektiv und bewertet eine Stellung nur anhand der „Fakten". Gegen einen Menschen braucht er nur auf Fehler zu warten und diese dann ausnutzen. Diese Fehler ergeben sich oft zwangsläufig, resultierend aus dem zweiten Vorteil des Computers: der Rechenstärke. Innerhalb seines Horizontes übersieht der Computer nichts; er ist im Gegensatz zum Menschen imstande, taktische Zugfolgen auf 20 Halbzüge (also 10 Züge) vorauszuberechnen. Auch findet der Computer viele versteckte Zugfolgen, die der Mensch aufgrund seiner hohen Selektivität nicht weiter betrachtet hat, da sie ihm auf den ersten Blick sinnlos erschienen.

1.3 Ist es noch möglich, den Computer zu besiegen?

Mittlerweile gilt es als sicher, dass der beste Schachspieler gegen die beste Software, auf der besten Hardware spielend, nicht mehr mithalten kann. Aber natürlich gibt es immer noch schwächere Soft- und Hardware, die es dem Menschen möglich machen, eine Engine zu besiegen. Aus den oben skizzierten Schwächen lässt sich leicht ableiten, auf welchem Weg dies am besten zu bewerkstelligen ist.
Allgemein sollten taktische Stellungen vermieden werden, damit der Computer nicht zum Ausspielen seines gewaltigen Rechenvorteils kommen kann. Stattdessen sollten geschlossene Stellungen angestrebt werden, wo aufwendiges Manövrieren und langfristiges Planen erforderlich ist. Wie später noch deutlicher beleuchtet wird und schon angeklungen ist, stellt gerade langfristiges Planen eine große Schwäche des Computers dar.

Kramnik benutzte in seinen Computermatches 2002 und 2006 gegen *Deep Fritz 7/10* immer die gleiche Strategie: Er wählte geschlossene Eröffnungen, strebte schnellen Damentausch an und versuchte möglichst schnell ins Endspiel überzuleiten. Der Damentausch bewirkte, dass weit weniger taktische Manöver möglich sind und viel eher langfristige Strategien greifen. Gleiches gilt für das Endspiel, wo es selten aufgrund des reduzierten Materials zu konkreten taktischen Motiven kommt, sondern in erster Linie logisches und strategisches Denken erforderlich ist. So ist auch das Endspiel nach wie vor der größte Schwachpunkt der Schachcomputer, wenn auch Schwachpunkt zu relativieren ist.

Aber selbst für fortgeschrittene Vereinsspieler ist es noch möglich, ein PC-Spitzenprogramm auf einem handelsüblichen Rechner zu besiegen, wie ich erst kürzlich feststellen musste, als *Shredder 9* in einer eigenen Schnellpartie lange Zeit einen Freibauern völlig unterschätzte und ihn erst ernst nahm, als es schon zu spät war. Diese Schwächen sind aber nur vereinzelt; tendenziell sind die Programme abgesehen von den genannten krassen Fehleinschätzungen natürlich dem Durchschnittsspieler klar überlegen.

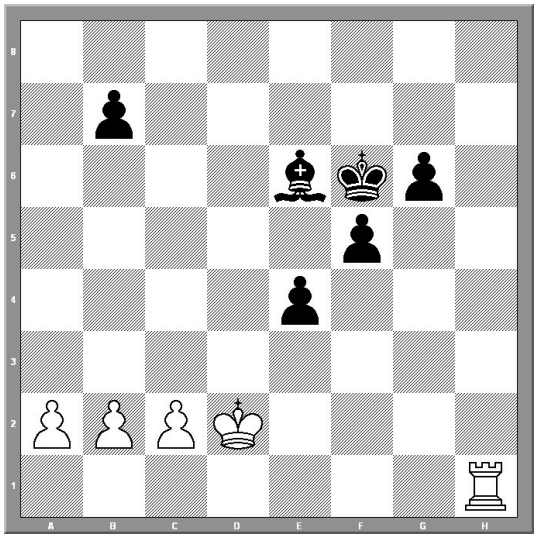
Wie spätestens seit Kramniks 2–4-Niederlage gegen *Deep Fritz* 2006 als sicher gilt, reichen diese Strategien nicht mehr um ein Spitzenprogramm tatsächlich zu besiegen. Kramnik erreichte mehrere Male eine bessere Stellung, aber bis zu einem Sieg ist war es ein langer Weg, den kein Mensch mehr zu einem Sieg führen kann, weil dies bedeuten würde, mehr als zwanzig Züge lang perfekt zu spielen.

1.4 Blinde Flecken der Schachprogrammierung

Vorweg möchte ich für eine ausführlichere Betrachtung des Abschnittthemas auf das Buch „How to use computers to improve your chess“ von Christian Kongsted, genaueres ist im Quellenverzeichnis einzusehen, hinweisen, an das ich mich in diesem Kapitel teilweise anlehne.

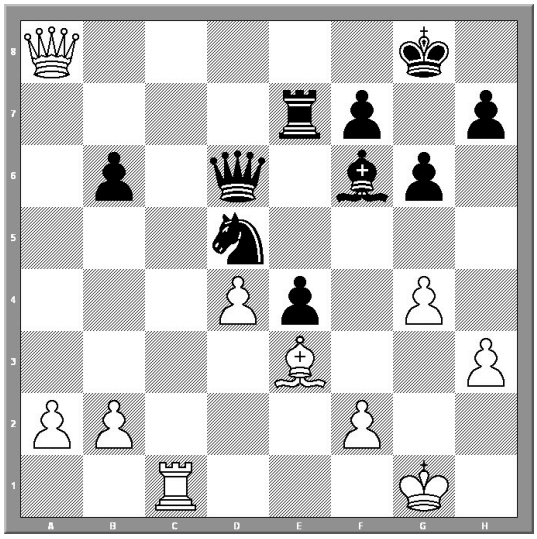
Der größte Teil, worin sich jedes kommerzielle Programm unterscheidet, ist wohl seine Bewertungsfunktion. In ihr wird ganz grob gesagt entschieden, weswegen welcher Zug wie gut ist. Zu diesem „intelligenten“ Part des Programms sind natürlich dem Programmierer alle Freiheiten gegeben; hier unterscheiden sich die guten von den sehr guten Programmen. Daraus ergibt sich auch, dass jedes Programm seinen ganz persönlichen Stil und damit einhergehend auch ganz individuelle Stärken und Schwächen hat, weil jeder Programmierer andere Techniken, Bewertungskriterien und Gewichtungen verwendet. Eine dieser immer noch vorhandenen, teils äußerst ausgeprägten Schwächen möchte ich nun im Folgenden beschreiben.

Eine der größten generellen Schwächen der Computer ist der so genannte „Horizont-Effekt“. Dieser besteht darin, dass der Computer natürlich wie der Mensch nur eine gewisse Maximaltiefe erreichen kann und alles, was hinter ihr geschieht für ihn unbekannt ist. Des Menschen Vorteil ist aber, dass er dieses Unbekannte weit besser einschätzen und viel ökonomischer und zweckmäßiger festlegen kann, wo er seine Variantenberechnung beendet und wo er noch weiterrechnet.
Diesen Effekt möchte ich an einem zwar schon alten, aber nichtsdestotrotz sehr instruktiven Beispiel verdeutlichen.



In der vorliegenden Stellung ist es für jeden halbwegs bewanderten Spieler selbstverständlich und selbst für Anfänger nachzuvollziehen, dass der schwarze Läufer auf e6 (für Probleme mit der Notation hat, bitte ich, im Sachregister unter Notation nachzulesen) nach 1...Lxa2 2.b3 gefangen ist (er kann nirgendwo hin mehr entweichen, ohne geschlagen zu werden) und durch Ta1 oder Kc3/c1-b2 sehr simpel gewonnen werden kann. Vor 10–15 Jahren hätten viele der damaligen Spitzenprogramme den Bauern nichtsdestotrotz ohne jegliche Skrupel genommen. Der Grund hierfür ist das eingangs skizzierte Horizontproblem. Schwarz kann durch viele Schachgebote das Schlagen des schwarzen Läufers bis hinter den Horizont verschieben, sodass dessen Eroberung, obwohl für jeden Menschen leicht ersichtlich, hinter den Rechenhorizont des Programms verschoben wurde. Nach beispielsweise 2...f4 3.Ta1 e3+ 4.Ke2 f3+ 5.Kxe3 Lxb3 6.cxb3 wird der Läufergewinn bis auf Tiefe 10 hinausgeschoben, sodass viele der damaligen Programme ihn noch nicht erkennen konnten.

Ein anderes illustratives Beispiel zeigt eindrucksvoll, wie sehr sich Mensch und Computer unterscheiden: Der Programme größte Stärke kann gegen den Menschen zu einer großen Schwäche werden. Dies möchte ich anhand der nächsten Partie erläutern.

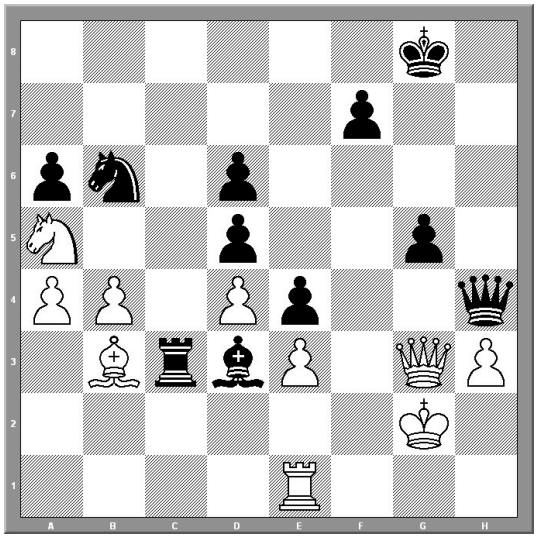


Diese Stellung ergab sich zwischen *Duchess* und *Kaissa* auf der 2. Computerschachweltmeisterschaft 1977. „Vollkommen unerwartet für das aus einigen Hundert Computerfachleuten und Schachmeistern bestehende Publikum opferte *Kaissa* mit 34...Te8 den Turm und verlor nach etwa 15 weiteren Zügen die Partie. [...] Offensichtlich konnte nur ein grober Programmierfehler vorliegen.

Am nächsten Morgen gaben die Entwickler von *Kaissa*, Arlasarow und Donskoj das Ergebnis ihrer nächtlichen Untersuchungen bekannt: Nach 34...Kg7? 35.Df8+! Kxf8 36.Lh6+ Kg8/Lg7 37.Tc8 nebst Matt in zwei Zügen.“[5] Es stellte sich heraus, dass das Programm tatsächlich den besten Zug gespielt hatte, nämlich den einzigen, der das sofortige Matt verhinderte. Somit hatte es weiter in die Stellung gesehen als alle versammelten Schachexperten, darunter auch Exweltmeister Botwinnik! Doch diese große Stärke wäre in einer Partie gegen einen Menschen eine krasse Schwäche geworden, da die Gewinnführung nach dem Turmverlust für Weiß natürlich viel einfacher ist, als nach 34...Kg7 das Matt zu finden.

Heutzutage gibt es natürlich viel schnellere Computer und damit eine tiefere Vorausberechnung und weiterhin verbesserte Suchtechniken, sodass solche extremen Beispiele nur noch selten anzutreffen sind. Ein Beispiel der letzten Zeit ist *Fritz*–Ponomarjow, Bilbao 2005.

[5] Posthoff/Reinemann: Computerschach – Schachcomputer, Akademie-Verlag Berlin 1987, Seite 122



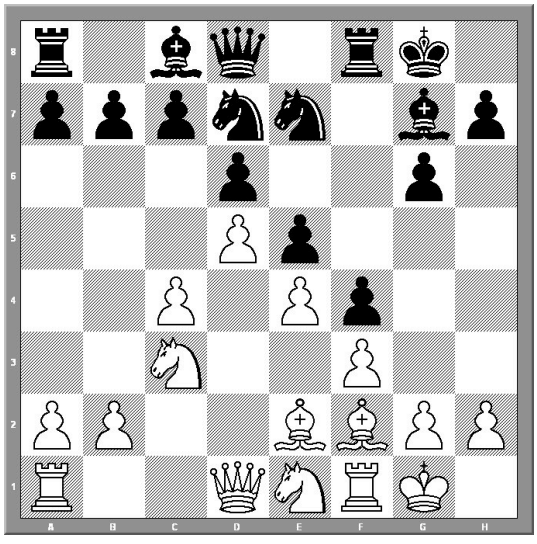
Fritz als Schwarzer spielte Lc2 und Ponomarjow konnte sein Glück kaum fassen: Für *Fritz* nicht erkennbar konnten Ponomarjow und sämtliche Experten im Pressezentrum mit Sicherheit sagen, dass der weiße auf der a-Linie entstehende Freibauer die Partie für ihn gewinnt. Doch da der Umwandlungsprozess eines Freibauern äußerst langsam vonstatten geht, wird er sehr weit hinter den Rechenhorizont verschoben, sodass die die Problematik oft sehr schwer einschätzen können. Ein amüsanter Kommentar hierzu stammt von Chrilly Donninger, dem Programmierer von Hydra, dem weltbesten Schachcomputer: „Die einzige Figur mit unklarem Materialwert ist der Freibauer. Mit Freibauern haben die Programme daher auch heute noch ihre liebe Not. Siehe zum Beispiel die Partie *Fritz* – Ponomarjow, Bilbao 2005. *Fritz* hat die Partie in klar gewonnener Stellung vergurkt. weil es noch Material abräumen wollte und einen möglichen Freibauerndurchbruch von Ponomarjow nicht durchrechnen konnte. Die anwesenden GMs [Großmeister – A.F.] sahen hingegen sofort, dass das für *Fritz* in die Hose geht. Genau genommen sahen sie es sofort, aber ihr Glaube an die taktische Unfehlbarkeit der Programme ist inzwischen so groß, dass sie nach einer taktischen Pointe von *Fritz* Ausschau gehalten haben. Es gab aber keine, die Umwandlung war nur jenseits von *Fritz*ens Horizont.

Umgekehrt schätzte in Topalow gegen Hydra, Bilbao 2004, Hydra einen eigenen Bauern auf d3 viel zu hoch ein. Der Freibauer wurde, obwohl langfristig unhaltbar, mit Zähnen und Klauen verteidigt. Hydra versaute sich völlig die Stellung. Ich war froh, als Topalow den Bauern endlich hatte.“[6]

[6] C. Donninger: “Nur 64 Felder“, in: Kaissiber 23, Dreier Verlag, Seite 7

Dies waren aber auch sehr extreme Beispiele; heutzutage schafft der Horizonteffekt mehr positionelle Probleme als taktische Schnitzer. Hier wird das Problem des langfristigen Planens deutlich. Heutige Programme erreichen unter Turnierbedingungen (2h/40 Züge) durchschnittlich bis zu 18 Halbzüge im Mittelspiel. Manöver, die innerhalb des Horizontes liegen, sehen für den menschlichen Spieler oft wie ein guter Plan aus, für den Computer sind es aber nichts als einzelne Züge bzw. Zahlen ohne Zusammenhang. Daraus ergibt sich, dass das Planen für einen Computer mit der gängigen Programmiertechnik nicht möglich ist. Manöver, die aufgrund von taktischen Problemen (Figurenangriffe, die erwidert werden müssen) jenseits des Rechenhorizontes verbannt werden, können somit nicht erkannt werden. Es gab verschiedene Ansätze dieses Problem zu lösen, aber gänzlich zufrieden stellend konnte die Sache noch nicht geregelt werden.

Eine daraus resultierende große Schwäche der Programme von damals und heute, wenn auch nicht mehr so stark, ist das Behandeln von geschlossenen Positionen, wenn konkrete taktische Schlagabtäusche fast keine Rolle bei der Stellungsbewertung spielen. Nehmen wir nur ein Beispiel einer äußerst populären Theorie-Stellung der Königsindischen Verteidigung.



Der mit großem Abstand meistgespielte Zug in der Stellung ist 12...g5. Alternativen sind 12...h5 und 12...Sf6. Alle diese Züge haben gemein, dass sie planen, mithilfe des Vormarsches der g und h-Bauern einen Angriff auf die weiße Königsstellung vorzunehmen. Bei 12...h5 und 12...g5 geschieht dies unverzüglich, 12...Sf6 bereitet den Vorstoß vor. Dieser

Plan ist für Menschen äußerst einfach zu verstehen, auch wenn die Konsequenzen des Angriffs weit mehr als 20 Halbzüge auf sich warten lassen. Aber da das Zentrum abgeriegelt und sicher ist, erscheint es sinnvoll auf dem Königsflügel anzugreifen, wo die schwarzen Figuren schon gut zum Einsatz gruppiert sind.

Programme hingegen werden sich niemals für diesen Plan entscheiden, mit Ausnahme, dass sie einen speziellen Bonus haben, in Stellungen mit geschlossenem Zentrum die Königsflügel-Bauern vorzuziehen. Die Programme erkennen nur die unmittelbare Folge, dass ihr eigener Bauernschutz um den König zerstört wird, dass langfristig auch der gegnerische aufs Korn genommen wird, können sie nicht erkennen.

Ich habe verschiedene Versionen der derzeit besten Programme diese Stellung mit folgendem Ergebnis 1 Minute lang analysieren lassen:

Fritz 8:	12...h6
Bright 3.0a:	12...b6
Cyclope 1.0:	12...b6
Fruit 2.3.1:	12...Kh8
Naum 2.0:	12...b6
Rybka 2.2:	12...a5
Spike 1.2 Turin:	12...b6
Strelka 2.0:	12...a5
Shredder 9:	12...g5
Glaurung 2.1:	12...b6

Auffallend ist, dass nahezu alle Programme beabsichtigen, am Damenflügel vorzugehen, obwohl die Aktionsplätze für beide Seiten eindeutig gegeben sind: Weiß hat deutlichen Raumvorteil am Damenflügel, Schwarz am Königsflügel. Die einzigen Ausnahmen sind *Fruit* und *Shredder*. *Fruit* wählt 12...Kh8 aus dem einfachen Grund, dass es erkennt, dass der König auf der offenen Diagonale a2-g8 steht und nicht einschätzen kann, dass diesem da gar keine Gefahr droht. *Shredder* spielt zwar 12...g5, setzt jedoch im nächsten Zug mit 13...b6 fort und verbindet also den richtigen Zug mit der falschen Idee. Folglicherweise gelingt es keinem der Programme, diese Stellung richtig zu behandeln. Dass dies auch in ganzen Partien mit längerer Bedenkzeit gegen Menschen passieren kann, will ich anhand der Partie Kasparow – *X3D Fritz*, New York 2003 (siehe Anhang) erläutern. Kasparow schickte seine Bauernphalanx gegen den gegnerischen Damenflügel, sodass er *Fritz* Schritt

für Schritt einschnürte, ohne dass das Programm Gefahr witterte. Der Effekt lag weit hinter seinem Rechenhorizont und so sah es sich selber im Vorteil. Während Kasparow äußerst zielstrebig seinen Plan verfolgte und völlig normale Züge zog[7], zog *Fritz* ohne erkennbaren Sinn hin und her. Es war allzu deutlich, wie stark der Mensch dem Computer in dieser Stellung überlegen war.

1.5 Status der Schachprogramme heute und Nutzungsmöglichkeiten

Mittlerweile sind Schachcomputer unentbehrlich für jeden Spitzenspieler geworden. Einerseits benötigt man die riesigen Datenbanken für eine profunde Partievorbereitung, da man so die Möglichkeit hat, sich alle Partien des jeweiligen Gegners anzusehen und auszuwerten. Außerdem nimmt der Computer sehr viel Arbeit bei der Analyse ab, sodass auch manche Großmeister heutzutage mehr den Computer als ihren eigenen Kopf zur Vorbereitung nutzen. Beispielhaft mögen dies einige Zitate des deutschen Spitzenspielers Jan Gustafsson, dem ich auf keinen Fall Unkreativität oder Ähnliches hiermit unterstellen will, illustrieren. Nach einer Neuerung kommentierte er: „Gefunden wurde der Zug durch den üblichen kreativen Prozess, den Computer mit allen halbwegs sinnvollen Zügen zu füttern und abzuwarten, ob irgendwas funktioniert.“[8] Eine starke Neuerung des Gegners kommentierte er: „Diesen Zug hatten wir auch auf dem Brett, aber wir widmeten ihm nicht genügend Zeit. Der Computer zeigt hier gleich ‚gut für Weiß‘ an. Damit war ich zufrieden und dachte: ‚16...Db6 spielt sowieso kein Mensch‘...“[9] Auch sonst gibt er als Hauptreferenz immer sein Programm *Rybka* an: „Soweit hatte ich mir alles angeguckt. *Rybka* zeigt hier ‚plus 1‘ oder so an, weil Weiß beide Damenflügelbauern erobert“[10] oder „*Rybka* ist mit meinem Spiel in dieser Phase zufrieden, also bin ich es auch.“[11]

So geht es heute im Spitzenschach darum, eine möglichst gute Balance zwischen der Rechenstärke der Computer und eigener Kreativität zu finden, um die Programme möglichst effektiv einsetzen zu können. „Man darf der Kiste nicht blind vertrauen. Das Wichtigste ist und bleibt die gedankliche Arbeit am Brett. Es kommt immer darauf an, die goldene Mitte zu finden, die ihm hilft, die Rechenkraft des Computers so effektiv wie

---

[7] GM Joel Lautier, www.chessbase.de/nachrichten.asp?newsid=2605
[8] Schach 1/2008, Seite 8
[9] ebd., Seite 14
[10] ebd., Seite 14
[11] Schach 12/2007, Seite 26

möglich zu nutzen. Das ist nicht einfach, weil so eine Maschine nicht nur Gutes bringt, sondern auch der Kreativität des Schachspielers schaden kann“[12], Kramnik.

Es gibt auch Schachformen, die sich speziell diesem Aspekt des Sportes widmen: Das so genannte „Advanced Chess“ ist eine Schachform, bei der die Spieler während der Partie einen Computer konsultieren können und beim „Freestyle Chess“ sind sogar sämtliche Hilfsmittel erlaubt; letzteres wird über das Internet ausgetragen.

Weiterhin gibt es auch Fernschach, das ähnlich abläuft wie „Freestyle Chess“, nur dass die Partie nicht zeitnah ausgetragen wird, sondern mehrere Tage für die Antwort je nach Regularien zur Verfügung stehen. Wie stark hier der Computer genutzt wird, ist wohl von Spieler zu Spieler unterschiedlich; es gibt Fernschachspieler, die es immer noch als ihre heilige Pflicht sehen, alles selbständig zu erforschen und zu analysieren und andere passen sich dem Zeitgeist an und wählen auch den Computer als Analysemedium.
Gustafsson wiederum beschreibt folgendermaßen das Vorgehen in der heutigen Weltspitze: „Du gibst alle Züge, die halbwegs Sinn machen, in *Rybka* ein, guckst dir ein paar Varianten an und was interessant aussieht, untersuchst du näher. So läuft das. Im Brute-Force-System.“[13]

So ist es heute ein entscheidender Vorteil, eigene Ideen zu haben, die der Computer nicht findet und somit der Gegner nicht so einfach aufspüren kann. Der jetzige Weltmeister Anand meinte 2005 auf eine starke Neuerung von Topalow: „Keine leichte Sache, so etwas zu finden. Mit dem Computer funktioniert das normalerweise nicht. Das ist ein Problem des modernen Schachs an sich.“[14] Dass man seine neuen Ideen bequem zu Hause vorbereiten kann, während der Gegner alles am Brett finden muss, ist ein unschätzbarer Vorteil, der auf Spitzenniveau ganze Partien entscheiden kann.

Verschiedentlich ist es zu beobachten, dass Spitzengroßmeister auf dem Brett ihre Analysen abgleichen, die sie zu Hause mit dem Computer angefertigt haben und danach sich auf Remis einigen. Mit dem Computer sind nun Eröffnungsvarianten einfacher zu erlernen, sodass man sich viel leichter auf seinen Gegner vorbereiten kann, da kein gezieltes Expertenwissen und Karteiarchive mehr vonnöten sind.

---

[12] www.chessbase.de/nachrichten.asp?newsid=8242
[13] Schach 1/2008, Seite 12
[14] Schach 7/2005, Seite 15

Schlussendlich wird also durch den Computer vieles bequemer, aber man sollte nicht seinen Verstand durch ihn ersetzen lassen, es kommt immer auf die gute Synthese an, um ihn wirklich effektiv zu nutzen, da auch die Programme noch nicht allwissend sind.

1.6 Aufbau eines Schachprogramms

Um dem zweiten Teil dieser BeLL gut folgen zu können, möchte ich hier einen kleinen Einblick in den prinzipiellen Aufbau eines Schachprogramms, erstmals 1949 von dem bekannten Mathematiker Claude Shannon formuliert, geben.
Folgende Bestandteile gehören heutzutage zum Standard eines Programms:

1. Interne Brettrepräsentation:

Die interne Brettrepräsentation regelt die Kommunikation zwischen Eingabe und Verarbeitung. Sie ‚übersetzt' die Stellung in eine für das Programm verarbeitbare Datenform. Hierfür gibt es viele verschiedene Techniken, zwischen denen sich ein Programmierer entscheiden kann.
Die simpelste und naheliegendste, aber leider auch sehr uneffiziente Variante wäre eine 8x8-Matrix als Felddarstellung zu benutzen und jedes Feld, je nachdem was sich darauf befindet, mit einer Zahl zu kodieren (zum Beispiel 0 für ein leeres Feld, positive Werte für weiße und negative für schwarze Figuren).
Als effizienter und heutiger Standard hat sich jedoch das sogenannte „BitBoard" etabliert.[15] Dabei wird beispielsweise für jede einzelne Figur ein eigenes Register geschaffen, bei dem die Anzahl der Bits die Anzahl der Felder auf dem Schachbrett darstellt. Durch eine einfache UND-Verknüpfung lassen sich leicht die Wechselwirkungen zwischen den Figuren festhalten. Dem Vorteil der Effizienz steht aber der Nachteil der Unübersichtlichkeit gegenüber.
Für weitere Techniken bitte ich anderweitige Quellen konsultieren. Eine nähere Betrachtung von mir würde hier den Rahmen und Zweck sprengen.

2. Zuggenerator:

Der Zuggenerator ermittelt alle regelkonformen Züge und speichert sie in einer Liste ab. Zusätzlich registriert er die Züge des Gegners und anschließend wieder die des Programms usw., je nachdem wie rechenstark das Programm ist.

[15] Block: "Reinforcement Learning in der Schachprogrammierung", Seite 7

Aber leider ist das alles nicht so einfach, wie es sich anhört. Die Effizienz jedes Bestandteils und damit auch des gesamten Programms hängt stark davon ab, wie lange die Berechnungszeit für den jeweiligen Teils ist. Je kürzer diese dauert, desto tiefer kann das Programm rechnen und desto besser spielt es. Für ein anständiges Niveau spielen viele kleine Nuancen eine große Rolle, sodass man mit einfachen Techniken nur eine ziemlich geringe Spielstärke realisieren könnte.
Ferner hat der Zuggenerator auch die Aufgabe, die Züge des Gegners auf deren Korrektheit zu überprüfen, Schachgebote und Mattstellungen festzustellen.

3. Bewertungsfunktion:

Die Bewertungsfunktion beurteilt die Güte aller Zugfolgen, die vom Computer nach bestimmten, bei allen Programmen unterschiedlichen Kriterien berechnet wurden. Als Beispiele sollen hier das materielle Kräfteverhältnis beider Seiten und ferner positionelle Kriterien wie Bauernstruktur, Figurenbeweglichkeit, Zentrumsbeherrschung usw. genannt werden.

4. Zugauswahlprogramm:

Nachdem der Computer alle Züge bewertet hat, braucht er natürlich einen Algorithmus, durch den er möglichst effizient feststellen kann, welcher Zug der günstigste ist. Der Basis-Algorithmus für diesen Teil des Programms ist der so genannte MinMax-Algorithmus: Er geht von der Logik aus, dass jede Seite den für sich jeweils besten Zug spielt. Eine Zugfolge wird also nach der Stellungsbewertung beurteilt, die bei beiderseits bestem Spiel herauskommt.
Da dieser Algorithmus ohne Zusätze den kompletten Spielbaum durchsuchen müsste und damit sehr viel Zeit in Anspruch nehmen würde, versucht man ihn durch verschiedene Techniken abzukürzen, die es dem Programm ermöglichen, bestimmte Teile des Zugbaumes von vornherein als irrelevant auszuschließen und nicht weiter zu betrachten, um Zeit zu sparen.
In welcher Form der Zug letztendlich ausgegeben wird (Brettanzeige, vierstellige Koordinatenanzeige, Diodenanzeige eines Schachbrettes usw.) spielt bei der prinzipiellen Programmstruktur keine Rolle.

2. Teil: *Nexus* – ein ganz normales Schachprogramm?

2.1 Der Arbeitsprozess

Schon lange entstand bei mir der Wunsch, ein eigenes Schachprogramm zu schreiben. Als ich erfuhr, dass für die BeLL auch ein praktischer Teil erforderlich sei, machte ich meinen Wunsch zur Aufgabe und begann mit der Hilfe von Stefan Richter ein Schachprogramm in Java zu schreiben. Schon bald entschied ich mich aber doch dagegen, da Java wohl für Einsteiger recht geeignet ist, aber wohl kaum dafür, ein leistungsfähiges Schachprogramm auf die Beine zu stellen. Nach kurzer Überlegung entschied ich mich für C++. Dies hatte drei Gründe: Der Hauptgrund war selbstverständlich, dass es für die Schachprogrammierung dank seiner Effizienz sehr gut geeignet sind (viele Schachprogramme sind in C oder C++) und jeder Schachprogrammierer, den ich als externen Betreuer engagieren würde, könnte damit wahrscheinlich mehr anfangen als mit Java. Der zweite Grund war, dass ich selber auch über einige Kenntnisse dieser Programmiersprache verfügte, da ich in der achten Klasse ein Additum zu diesem Thema belegte, außerdem konnte ich Stefan Werner für diese Programmiersprache als Hilfe gewinnen.

Dann begann ich mit der Arbeit. Ich konnte schnell mit Stefan darin übereinkommen, dass wir uns sämtliche Teile und Techniken des Programms selber überlegen und so wenig wie möglich auf eventuell schon existierende Quelltext zurückgreifen wollten; das Hauptaugenmerk sollte mehr auf Verständnis als auf Qualität liegen. Dementsprechend dauerte der Prozess recht lange, wir hatten selbstverständlich nicht immer genügend Zeit, um uns darin zu vertiefen, da diese Arbeit sehr mühselig ist und viel Geduld sowie Konzentration erfordert.

Schon zu Beginn überlegte ich an Möglichkeiten, um tatsächlich eine Intelligenz zu schaffen, die völlig neuartig sein sollte. Sie sollte das können, wovon die Wissenschaft immer noch träumt: Sich selber erschaffen. Sie sollte selber Pläne und deren Umsetzung entwerfen und dies alles, ohne dass wir ihr vorher algorithmisch definiert haben „wenn die Konstellation so ist, nehme diesen Plan und so weiter“. Sie sollte selber die Strategie des Schachspiels entwickeln können, völlig ohne dass wir ihr eine Vorgabe machten.

Aber das Suchen blieb erfolglos. Die perfekte Strategie scheint es sowieso nicht geben zu können, denn bei dieser müsste es egal sein, wie tief das Programm rechnet, da dieser Algorithmus für jede Stellung ohne Vorausberechnung den besten Zug liefern müsste. Man kann sie nicht völlig ausschließen, aber zumindest ist sie recht unwahrscheinlich.

Daher entschied ich mich zunächst für die „kleine“ Lösung und strebte danach, ein möglichst voll funktionsfähiges Schachprogramm zu haben, was dem modernen Standard genügt. Dies dauerte seine Zeit, da sich Stefan nicht mit den normalen Programmiertechniken für die Zugarten der einzelnen Figuren anfreunden konnte und stattdessen eine elegantere Lösung suchte. Aber diese wurde gefunden und es scheint, dass diese Technik hier neu erfunden wurde.

Das Regelerlernen wurde schließlich nach langer Zeit geschafft und damit die langwierige Vorarbeit erledigt. Nach diesen Grundlagen konnte endlich die richtige Arbeit beginnen. Sie verlief in zwei Strängen: das Programm sollte intelligenter und schneller gemacht werden. Die Schnelligkeit wurde mit der Einführung der gängigen Zugsuchalgorithmen (Minimax, Alphabeta) erhöht. Die Intelligenz wurde zunächst mit der Einführung des Materials und der Beweglichkeit als Bewertungskriterium gesteigert. Dem schloss sich bald die Erstellung einer Transpositionstabelle an, die jeder Figur für jedes Feld eine bestimmte Punktzahl zuweist. Ein Springer im Zentrum des Brettes erhielt dadurch beispielsweise eine höhere Punktzahl als am Rand. Auch wenn dies zunächst eine Spielsteigerung möglich machte, musste bald eingesehen werden, dass diese keinesfalls von Dauer ist, da das Programm nun viel zu schematisch und statisch agierte und bewertete. Es richtete sich dadurch nie nach den Erfordernissen einer speziellen Stellung, sein einziges Ziel war, jede Figur auf einen unabänderlich von mir festgelegten „besten“ Platz zu stellen.

Als nächstes wurde immer ein wenig an der Zugvorsortierung gearbeitet; es ist für die Zugsuchalgorithmen entscheidend, in welcher Reihenfolge das Programm die Zugfolgen berechnet. Berechnet es schon gleich als Erstes den besten Zug ist es weit schneller fertig, als wenn es dies als letztes tut. Dadurch kann durch eine gute Vorsortierung auch ein Geschwindigkeitsgewinn erzielt werden. Außerdem wurde eine Technik eingeführt, die ermöglicht, während der Partie je nach der Berechnungszeit für einen Zug die Berechnungstiefe zu verstellen. Das Programm konnte nun von selber im Endspiel tiefer rechnen als in der Eröffnung, was sinnvoll ist, da das Endspiel durch weniger Figuren weniger Zugmöglichkeiten und damit weniger Berechnungszeit benötigt.

Doch bald musste ich mir eingestehen, dass genau das, was ich eigentlich nicht wollte, ein unintelligentes, aber schnelles Programm nun entstanden war. Ich sah nun den Zeitpunkt für gekommen, meine Vorstellung für die Intelligenz des Programms umzusetzen und sah mich nach einer guten Möglichkeit um, wobei ich bald auf die Diplomarbeit meines jetzigen externen Betreuers, Herrn Marco Block, aufmerksam wurde, in der er die

Möglichkeit des Reinforcement Learning in der Schachprogrammierung aufzeigte, was bedeutet, dass das Programm ihm vorgeschriebene Bewertungskriterien selber wichtet und damit von selber eine Approximation an die imaginäre ideale Bewertungsfunktion schafft.
Neben kleineren Entwicklungen wie der Umschreibung des Programms zur Nutzung des UCI-Protokolls und damit auch zur Einbindung in populäre Engines wie *Fritz* und einer Verfeinerung der Ruhesuche, versuchte ich im Folgenden konsequent, dieses Ziel umzusetzen. Nach einiger Zeit entstand dann auch eine solche Funktion, die aber noch nicht völlig zufriedenstellend arbeitete. Leider dauerte es nun eine gewisse Zeit, bis wir ermitteln konnten, wie die Funktion korrekt aussehen müsste, sodass wir nicht den Spielstärkezuwachs mithilfe des Reinforcement Learning gut messen konnten. Aber letztendlich war dies ja auch nicht unser Ziel, sondern inwiefern sich allgemeine Bewertungsparameter zur Bewertung eignen.

2.2 Reinforcement Learning in der Schachprogrammierung

Für den interessierten Leser empfehle ich an dieser Stelle einen Blick in die Studien- und Diplomarbeit meines externen Betreuers Herrn Block, der beide Arbeiten diesem Thema widmete und die mir im Folgenden als Hauptreferenzquelle dient.
Reinforcement Learning ist eine Lernstrategie, bei der der „lernende Agent selbst durch Ausprobieren und Rückmeldungen [...] die optimale Strategie, um Ziele seiner Umgebung zu erreichen“[16], findet. Für die Schachprogrammierung eignet sich hierbei besonders die sogenannte Temporal-Difference-Methode. Sie bewertet mithilfe einer Bewertungsfunktion die einzelnen Stellungen der Partie und schließt von deren Bewertungsänderungen auf eventuelle Fehler beziehungsweise Fehlbewertungen der vorangegangenen Züge.
Reinforcement Learning lässt sich in jedes moderne Schachprogramm einbauen und wird verwendet, um die Koeffizienten der Bewertungsfunktion durch „try and error“ selbst zu justieren. Dies ist die einzige Möglichkeit, die Bewertungsfaktoren objektiv zu setzen, da bei Setzung durch den Menschen stets sein subjektiver Geschmack mit hereinspielt.

2.3 Die Bewertungskriterien

Die Bewertungskriterien sind die Merkmale einer Stellung, die das Programm als positiv oder negativ für sich wertet. Dass von dieser Bewertung maßgeblich der Spielstil der Engine abhängt, erscheint logisch: Wenn eine Engine hohe Punkte auf eine sichere

[16] Block: “Reinforcement Learning in der Schachprogrammierung“, Seite 4

Königsstellung vergibt, wird es weit besorgter um seinen eigenen König sein, als eine Engine, die weniger Punkte darauf gibt.
Diese Bewertungsfunktion ist der Versuch, das menschliche Schachwissen einem Computer beizubringen. Das wäre alles relativ einfach, wenn nicht jeder Mensch eine andere Gewichtung der Kriterien hat und es keine objektiv beste gibt, sowie dass kein Mensch diese von ihm unterbewusst vorgenommene Gewichtung in einer linearen Funktion auszudrücken vermag, da die Erfahrung, Motiverkennung und Intuition eine zu große Rolle im menschlichen Denken spielt, als dass man es einfach übertragen könnte.
Schon bei der Bewertungskriterienbeschreibung gibt es Schwierigkeiten: Was bedeutet Königssicherheit? Was bedeutet Zentrumskontrolle? Dazu kommt das schon beschriebene Problem, dass das Programm, umso komplexer seine Bewertungsfunktionen sind, umso langsamer rechnet. Und da das Muster Königssicherheit, wöllte man es exakt beschreiben, äußerst komplex wäre, wird es meist nur vereinfacht wiedergegeben (es sei nur auf Fruit und seine Klone verwiesen, die die Königssicherheit danach bewerten, ob bei König auf g1 bzw. g8 Bauern auf g2 und h3 oder h2 stehen, was natürlich äußerst statisch ist).
Mein Ziel war es, Bewertungskriterien zu finden, die dem Computer möglichst selber die größte Bandbreite zur „kreativen Anwendung“ geben. Das bedeutet, dass der Computer keine starren Regeln bekommen sollte, sondern die übergeordneten Regeln, die die speziellen Regeln erklären und ihm somit auch erlauben, die Ausnahmen der speziellen Regeln zu erfassen.
Doch was sind solche allgemeinen Regeln? Zunächst wurde ich auf die bloße Anzahl der Züge, die jede Seite zur Verfügung hat, aufmerksam. Es erscheint logisch, dass die Seite im Vorteil ist, die mehr Möglichkeiten zum Ziehen aufweist, schon allein aus dem Grund, dass das Spiel verloren ist, wenn man nicht mehr ziehen kann, dass heißt, man Matt gesetzt ist (Patt wird hier ausgeklammert). In Fachkreisen wird dieses Kriterium Beweglichkeit oder Mobilität genannt.
Aber es gibt genügend Ausnahmen dieser Regel, denn nach ihr sind Bauern, die nicht mehr ziehen können, was öfters der Fall ist, wertlos, es erscheint zweckmäßig, die Dame zu Beginn des Spiels zu ziehen, wobei jeder Schachlehrer stets mahnend den Mittelfinger hebt außerdem würde das Programm versuchen, dem König beweglich zu machen. Dass der König an sich durchaus als aktive Figur zu gebrauchen ist, hat schon der erste Schachweltmeister, Wilhelm Steinitz (1836–1900), festgestellt, aber natürlich muss er zunächst geschützt werden. Mir fiel aber rasch auf, dass diese Ausnahmen der Beweglichkeitsregel alle

mit sich bietenden Angriffsmöglichkeiten zu tun haben. Ein unbeweglicher Bauer hat durchaus Wert als Sperrstein, solange er nicht gewinnbringend angegriffen werden kann, eine Dame sollte nicht zu früh gezogen werden, weil sie leicht angegriffen und zurückgetrieben werden kann und man dadurch wertvolle Tempi verlieren würde und dass der König auch nicht angegriffen werden sollte, ist logisch.

Froh stellte ich fest, dass ich eine zweite allgemeine Regel gefunden habe und suchte nach weiteren. Um an die Grundlagen jeder Regel zu kommen, sollte man sich stets nach ihrem Sinn fragen. Was ist der Sinn davon, die Figuren zu zentralisieren? Der springende Punkt ist, dass dieser Brettteil die größte Flexibilität bringt, das heißt, dass alle Teile des Brettes verhältnismäßig schnell erreicht werden können und damit auch gut kontrolliert werden können. Außerdem werden die Zugmöglichkeiten der gegnerischen Figuren besonders wirksam eingeschränkt und die meisten eigenen Figuren entwickeln im Zentrum ihre größte Beweglichkeit. Mein Gedankengang war nun der, dass man die Bewertungsfunktion nun wesentlich einfacher, schneller und gleichzeitig intelligenter machen könnte, wenn man tatsächlich nur die wesentlichen Bestandteile, aus denen sich der Rest ergibt, nimmt. In diesem Fall ist dies die Brettkontrolle und eben die Beweglichkeit. Was macht nun die Brettkontrolle aus? Es geht wohl nicht darum, so viele Felder wie möglich zu kontrollieren, sondern die für jede Position entscheidenden. Das erscheint logisch: Beim Königsangriff geht es darum, die Felder nahe des gegnerischen Königs zu kontrollieren, beim Flügelangriff, dort ein Kräfteübergewicht zu besitzen und ein starkes Zentrum zu haben. Es kommt also immer auf die Kontrolle bestimmter Felder an. Wie kann man aber algorithmisch die Wichtigkeit der Kontrolle bestimmter Felder in jeder Stellung bestimmen?

Dies ist eine der vielen Fragen, die ich mir im Laufe der Arbeit stellte und die wohl auch entscheidend für das Schachverständnis sind.

Nachdem ich sehr lange darüber nachdachte, kam ich zu dem Schluss, dass das Schachspiel aus Figuren und deren Wechselwirkungen zueinander aufgebaut ist. Diese Wechselwirkungen zwischen den Steinen sind angreifen, verteidigen und schlagen für den einzelnen Stein seine normale mögliche Zuganzahl. Jedes Strategem muss sich also auf diese Bestandteile zurückführen lassen, sodass ich mit dieser Arbeit (meines Wissens nach das erste Mal) völlig auf spezielle Evaluierungen verzichtete und ausschließlich diese Kriterien verwendete: Material (schlagen), Anzahl der angegriffenen Figuren, Anzahl der verteidigten Figuren und Anzahl der möglichen Züge (ohne Schlagzüge). Mein Ziel lautete, ob diese Methode sich tatsächlich eignet wie es scheint, oder zu Recht nicht verwendet wird.

2.4 Temporale Differenz in der Schachprogrammierung

Meine zweite Idee, um das Programm ein wenig intelligenter zu machen, war, die sogenannte Temporale Differenz als Bewertungstechnik zu verwenden. Neben den allgemeinen Untersuchungen zu den Evaluierungskriterien sollte das Programm also auch noch lernfähig werden, was mit sehr viel Mühe verbunden ist.

Die Idee der Temporalen Differenz ist es, den Unterschied der Stellungsbewertungen nach jedem Zug zu minimieren. Diese Idee folgt dem Gedanken, dass die perfekte Bewertungsfunktion in der Schachprogrammierung schon bei Beginn des Spiels das Ende korrekt vorhersagen müsste und bei bestem Spiel des Gegners ihren Wert nicht mehr ändert.

So wird also die im letzten Zug ermittelte Stellungsbewertung durch die im nächsten Zug neu berechnete, nun tiefere Bewertung aktualisiert. Die Temporale Differenz versucht nun, diese Aktualisierungen möglichst gering zu halten. Dadurch kann sie die Zustandswerte optimieren, ohne ein Spielziel und die komplette Umwelt (alle möglichen Zugfolgen) zu kennen, was für die Schachprogrammierung sehr nützlich ist, da man mit ihr die Spielziele in Form der Bewertungskriterien testen kann, ohne sie festzulegen, da das Schachverständnis noch nicht so weit ist, um diese endgültig und definitiv zu benennen.

Im Folgenden möchte ich versuchen, die Aktualisierungsformel des TD-Algorithmus' zu erklären:

$$\omega := \omega + \alpha \sum_{t=1}^{N-1} \nabla J'(x_t, \omega) \left[\sum_{j=t}^{N-1} \lambda^{j-t} d_t \right]$$

Wie man sich sicher denken kann, ist die Temporale Differenz d_t die Differenz der Stellungsbewertung J' der Stellung zum Zeitpunkt x_t und zum Zeitpunkt x_{t+1}. Sie ermittelt also den Wert, um wie viel sich die errechnete Bewertung von der vorhergehenden zur nächsten Stellung verbessert bzw. verschlechtert hat.

Demnach lautet die Formel der Temporalen Differenz:

$$d_t := J'(x_{t+1}, \omega) - J'(x_t, \omega)$$

Unsere Bemühung ist nun, die Bewertungsfunktion so zu verändern, dass diese Temporale Differenz gegen Null konvergiert. Dies folgt dem Gedankengang, dass die perfekte Bewertungsfunktion, an die wir uns annähern wollen, schon am Anfang des Spieles bestimmen kann, ob es bei perfektem Spiel gewonnen oder verloren wird, die Temporale Differenz ist bei perfektem Spiel der Partie also Null, da sich die Bewertung nie ändert.

Mit diesem Wissen können wir schon den hinteren Teil der Formel verstehen. Er besteht daraus, dass sämtliche Temporale Differenzen für den Rest der Partie (Rest von dem vorhergehenden Teil der Formel) bis zum vorletzten Zug N-1 aufsummiert werden. Das λ^{j-t} hat dabei folgende Bewandnis: Es kontrolliert über einen beliebig festzusetzenden Parameter λ, wie stark die Temporalen Differenzen abhängig von ihren Zeitpunkt gewichtet werden. Dies folgt der Erkenntnis, dass nicht jede Temporale Differenz gleich stark gewichtet werden soll, da häufig erst zum Ende der Partie hin erkannt wird, dass man schlechter steht. Daher wird durch den Parameter λ reguliert, wie stark die Wichtung zum Ende der Partie hin zu- oder abnimmt. Das j ist der Startpunkt von der Laufvariable t; j-t liefert also immer die Differenz, um wie viel t vorangeschritten ist.

Das $\nabla J'(x_t, \omega)$ liefert den Gradienten. Er gibt grob gesagt an, was sich an dem bestehenden Bewertungsvektor ändert, während die Temporale Differenz angibt, um wie viel es sich ändert. Wenn sich die Bewertung eines Kriteriums im Vergleich zur Vorgängerstellung verbessert hat, wird der entsprechende Koeffizient des Kriteriums durch den Gradienten vergrößert, bei Verschlechterung verkleinert.

2.5 Die Beobachtungen

Da die Temporale Differenz erst spät korrekt implementiert werden konnte, liegen mir zum Abgabezeitpunkt noch keine aussagekräftigen Ergebnisse vor. Andere Experimente haben aber ergeben, dass nach 308 Partien eine beachtliche Steigerung von 500 Elopunkten (Wertungspunkte) bei dem Programm KnightCap[17] erzielt werden konnte. Selbstverständlich wird dieses Projekt nach der Abgabe der BeLL weiterlaufen, sodass ich hoffe, entsprechende Ergebnisse nachträglich zu erhalten.

Für mein eigentliches Untersuchungsthema, die Möglichkeiten des Einsatzes von allgemeinen Bewertungsparametern, sind die Beobachtungen weniger gut ausgefallen. Viele positionelle Probleme ließen sich nicht erfassen, was hauptsächlich daran lag, dass nicht tief genug gerechnet werden konnte, um die „Auswüchse“ der allgemeinen Grundlagen zu verstehen.

Das bedeutet, dass beispielsweise ein Freibauer zwar, wie ich mir dachte, natürlich durch Materialgewinn und Mobilitätsgewinn als schlecht eingestuft werden kann, aber dafür nicht

[17] M. Block: “Reinforcement Learning in der Schachprogrammierung”, Studienarbeit 2003

richtig erkannt werden konnte, da das Programm mit den bisherigen Optimierungen noch nicht über durchschnittlich 7 Halbzüge hinaus rechnen kann.
Ich denke aber, dass für die Schachforschung und –wissenschaft dieser neue Gedankenansatz dennoch interessant sein dürfte, da sich durch die vier genannten Bewertungskriterien tatsächlich sämtliche Phänomene erklären lassen.
Bei entsprechender Technik würde ich mich wundern, diese These nicht bestätigt zu sehen, auch wenn der Nachweis natürlich ausbleibt. Aber der interessierte Leser ist hiermit zum Ausprobieren aufgefordert!
So hoffe ich denn, mit dieser Arbeit vielleicht dem Einen oder Anderen dieses schöne Spiel mit seinen Facetten ein wenig näher gebracht zu haben und auch für begabte Spieler und Programmierer ein paar neue Gedankenansätze geliefert zu haben.

Kasparow – *X3D Fritz*, New York 2003

1. Sf3	Sf6	24. Kc1	Td8
2. c4	e6	25. Tc2	Sbd7
3. Sc3	d5	26. Kb2	Sf8
4. d4	c6	27. a4	Sg6
5. e3	a6	28. a5	Se7
6. c5	Sbd7	29. a6	bxa6
7. b4	a5	30. Sa5	Tdb8
8. b5	e5	31. g3	Lg5
9. Da4	Dc7	32. Lg2	Dg6
10. La3	e4	33. Ka1	Kh8
11. Sd2	Le7	34. Sa2	Ld7
12. b6	Dd8	35. Lc3	Se8
13. h3	O-O	36. Sb4	Kg8
14. Sb3	Ld6	37. Tb1	Lc8
15. Tb1	Le7	38. Ta2	Lh6
16. Sxa5	Sb8	39. Lf1	De6
17. Lb4	Dd7	40. Dd1	Sf6
18. Tb2	De6	41. Da4	Lb7
19. Dd1	Sfd7	42. Sxb7	Txb7
20. a3	Dh6	43. Sxa6	Dd7
21. Sb3	Lh4	44. Dc2	Kh8
22. Dd2	Sf6	45. Tb3	
23. Kd1	Le6	1 – 0	

Sachregister

Abtauschen/Abtausch: Siehe unter Tauschen/Tausch

Bauer: Kleinster Stein auf dem Schachfeld. Er zieht jeweils ein Feld vorwärts, außer in der Grundstellung in der er zwei Felder ziehen kann. Wenn er den gegnerischen Brettrand erreicht, muss er durch eine Figur außer König und Bauer ersetzt werden. Sein Wert ist eine Bauerneinheit.

Bauernstruktur: Typisierung der Anordnung von Bauern

Bedenkzeit: Festgelegte Regelung zum Zeitverbrauch während einer Schachpartie. Bei Überschreitung der Bedenkzeit wird die Partie als verloren gewertet.

Beweglichkeit: Ein Bewertungskriterium, dass ausdrücken soll wie beweglich eine Stellung für eine Farbe ist. Häufig zählt man hierzu die Anzahl der Zugmöglichkeiten

Dame: Nach dem König die höchste Spielfigur. Kann diagonal und gerade beliebig weit ziehen. Ihr Wert wird auf neun Bauerneinheiten geschätzt.

Damenflügel: die linke Bretthälfte, auf der die Dame steht

Decken: Eine Figur eigener Farbe mit einer anderen eigenen Figur überdecken, sodass man im Falle eines gegnerischen Schlagens zurückschlagen könnte

Diagonale: Die von den gleichfarbigen Feldern gebildeten Schrägen auf dem Schachbrett.

Endspiel: Allgemeine Bezeichnung für die Phase der Schachpartie, wenn nur noch so wenige Figuren auf dem Brett sind, sodass der König selber aktiv ins Spielgeschehen eingreifen kann

Engine: Bei einem Schachprogramm der Teil, der tatsächlich die Zugfolgen berechnet und bewertet. Viele Programme gibt es nur als Engines, das heißt ohne Benutzeroberfläche, sodass man ein anderes Programm zu ihnen braucht, um mit ihm spielen zu können.

Eröffnung: Erste Phase einer Schachpartie. In ihr geht es darum, seine Figuren möglichst zweckmäßig für den Rest der Partie aufzustellen, zu entwickeln.

Freibauer: Bauer, der von gegnerischen Bauern völlig unbehindert zur Umwandlung schreiten kann. Er stellt oft einen sehr großen Vorteil als Druckmittel dar, da sein Voranschreiten nur von höherwertigen Figuren als Bauern verhindert werden kann und nicht selten schafft er es tatsächlich zur Umwandlung

geschlossene Linie/Diagonale/Position: Eine Linie oder Diagonale heißt geschlossen, wenn sie durch Bauern verstellt wird, die als Sperrsteine für einen direkten Angriff auf das gegnerische Lager wirken. Eine Stellung mit vielen geschlossenen Linien oder

Diagonalen bezeichnet man auch als geschlossen. In ihr entscheidet langfristiges Planen von Strategien, da konkrete taktische forcierte Zugfolgen nicht viel existieren.

Grundstellung: Anfangsstellung

Halbzug: Bezeichnung für einen einzelnen Zug von Weiß und von Schwarz. Gegensatz zu Zug

König: Wichtigste und größte Figur. Seine Eroberung entscheidet über Gewinn und Verlust einer Partie

Königsangriff: Angriff auf die gegnerische Königsstellung

Königsflügel: Die rechte Bretthälfte, auf der der König steht

Königssicherheit: Ein Bewertungskriterium, was angibt, wie sicher der König vor eventuellen Angriffen geschützt ist

Läufer: Längere, dünne Spielfigur, die nur diagonal ziehen kann. Somit kann ein Läufer also nur die Hälfte aller Felder eines Schachbrettes betreten, da er nur auf die Felder einer Farbe ziehen kann

Linie: Die vertikalen Felder des Schachbretts vom Spieler aus gesehen

Material: die Schachfiguren aus dem quantitativen Aspekt

Matt: Wenn der König in Schach steht und nicht mehr daraus entweichen kann spricht man von einem Matt. In diesem Falle hat die mattsetzende Partei gewonnen, wodurch das Matt das Spielziel darstellt.

Mittelspiel: Die Phase zwischen Eröffnung und Endspiel. In ihr entscheidet sich meist der Ausgang der Partie

Neuerung: Der erste neue, bisher noch nicht gespielte Zug in einer Schachpartie

Notation: In der Schachsprache werden die Züge folgendermaßen notiert: Zuerst kommt ein Kürzel für den Stein, der gezogen wird. Im Deutschen sind das: K für König, D für Dame, T für Turm, L für Läufer und S für Springer. Der Bauer besitzt keinen eigenen Buchstaben. Danach folgt in der Kurznotation für einen Schlagzug ein x, für einen „normalen" Zug nichts und anschließend das Zielfeld. Dabei werden zuerst die Linien a bis h benannt und anschließend die Linien 1 bis 8. Ein Zug wäre demnach Dxa7, was bedeutet, dass die Dame eine Figur auf a7, also dem Feld vor dem in der linken oberen Ecke, schlägt.

Offene Linie/Diagonale/Stellung: Eine Linie oder Diagonale heißt offen, wenn auf ihr keine Bauern vorhanden sind, da sie freien Zugang zum gegnerischen Lager gewährt. Ihre Beherrschung bedeutet oft einen großen Vorteil. Eine Stellung mit vielen offenen

Linien oder Diagonalen bezeichnet man auch als offen. In ihnen entscheiden oft konkrete taktische Berechnungen und Zusammenstöße

Opfer: Preisgabe eines Steins ohne materiellen Ersatz bzw. Abtausch zweier Figuren von unterschiedlichen Tauschwert, mit dem Ziel, anderweitigen Vorteil zu erlangen.

Patt: Die am Zug befindliche Partei kann nicht mehr ziehen, weder mit König noch mit anderer Figur ohne im Schach zu stehen. Die Partie wird als Remis gewertet

Position: Stellung, Bezeichnung für die augenblickliche Situation auf dem Schachbrett

positionelles Spiel: Eine Strategie, die versucht durch Akkumulierung vieler kleiner langfristiger Vorteile die Stellung nach und nach zu verbessern, anstatt nach einer Möglichkeit zu sofortigen greifbaren Vorteil zu suchen

Reihe: die Felder auf einer vom Spieler aus gesehen vertikalen Geraden des Schachbretts

Remis: Unentschieden

Ruhesuche: Hilfsmittel zur teilweisen Verhinderung des Horizonteffektes. Ein Berechnung wird erst abgebrochen, wenn kein Schlag- oder Schachzug mehr möglich ist, um zu verhindern, dass eine Stellung positiv oder negativ gewertet wird, weil im letzten berechneten Zug eine Figur gewonnen/verloren wurde, obwohl im Folgezug diese zurückgewonnen werden kann

Schach: Bedrohen des gegnerischen Königs, sodass er geschlagen werden könnte. Ein Schachgebot muss pariert und kann auch nicht durch ein Gegenschach beantwortet werden

schlagen: Wegnehmen eines Steines

Springer: Pferd, die pferdeähnliche, drittkleinste Figur auf dem Brett. Der Springer besitzt als einzige Figur die Fähigkeit, andere Figuren zu überspringen, indem er zwei Felder nach links/rechts/vorne/hinten und ein Feld zur Seite springt und damit ein ‚L' beschreibt. Sein Wert wird auf 3 Bauern taxiert

Strategie: Die langfristige Zielsetzung und Planung. Strategische Stellungen sind oft geschlossene Stellungen, deren Charakter sehr dauerhaft sind und deswegen langfristiges Denken angebracht ist

Taktik: Im Unterschied zur Strategie beinhaltet Taktik die konkreten Maßnahmen, um eine Strategie umzusetzen. Eine taktische Stellung ist demgemäß eine, in der viel konkrete Berechnung eine Rolle spielt und weniger allgemeine, langfristige Strategien

Tauschen/Tausch/Abtauschen/Abtausch: Das gegenseitige Schlagen von Figuren

Tiefe: die Anzahl der vorausberechneten Halbzüge

Turm: Die turmähnliche, zweitkleinste Figur auf dem Brett. Er kann in alle Richtungen außer den Diagonalen beliebig weit ziehen und ist damit das Gegenstück des Läufers. Sein Wert wird auf 5 Bauern geschätzt

UCI-Protokoll: UCI (Universal Chess Interface) ist ein gängiges Kommunikationsprotokoll für Schachengines um mit der graphischen Benutzeroberfläche zu kommunizieren

Variante: Bestimmtes Abspiel, Zugfolge einer Schachpartie

Zentrum: Bezeichnung für die vier mittleren Felder des Schachbretts; d4, e4, d5, e4. Ihnen kommt oft durch ihre zentrale Lage eine große Bedeutung zu

Zug: Bezeichnung für eine Zugfolge, bei der jeweils Weiß und Schwarz einmal zieht. In anderem Zusammenhang auch für Halbzug verwendet

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Thema

Einsatz von allgemeinen Evaluierungsheuristiken in Verbindung mit der Reinforcement-Learning-Strategie in der Schachprogrammierung
Der Versuch einer künstlichen Intelligenz

selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, habe ich in jedem einzelnen Fall durch die Angabe der Quelle, auch der benutzten Sekundärliteratur, als Entlehnung kenntlich gemacht.

Meißen, den 17.12.2008 **Albrecht Fiebiger**

Quellen

Bücher:

Block: „Reinforcement Learning in der Schachprogrammierung“, Studienarbeit 2003

Block: „Verwendung von Temporale-Differenz-Methoden im Schachmotor FUSc#“, Diplomarbeit 2004

Botwinnik: „Meine neuen Ideen zur Schachprogrammierung“, Springerverlag 1982

Fiebiger: „Vergleich der Intelligenz von Mensch und Schachcomputer“, Vorstudie zur BeLL 2007

Hoffmann/Hoffmann: „Schach unter der Lupe“, Sportverlag Berlin 1986

Kasparow: „Meine großen Vorkämpfer“ Band 1, Edition Olms 2003

Kongsted: „How to use computers to improve your chess“, Gambit Verlag 2003

Kurz: „Musterverarbeitung bei der Schachprogrammierung“, Dissertation 1977

Posthoff: „Computerschach – Schachcomputer“, Akademie-Verlag Berlin 1987

Yazgac: „Schachcomputer: was sie wirklich können“, Beyer Verlag Hollfeld 1989

Zeitschriften:

Schach 07/05, Exzelsior Verlag GmbH

Schach 12/07, Exzelsior Verlag GmbH

Schach 01/08, Exzelsior Verlag GmbH

Kaissiber 23, Dreier-Verlag

Internetquellen:

de.wikipedia.org

www.top-5000.nl

ssdf.bosjo.net

page.mi.fu-berlin.de

www.schachcomputer.at

www.chessbase.de

Programme:

Nexus

Fritz 8

Bright 3.0a

Cyclope 1.0

Fruit 2.3.1

Naum 2.0

Rybka 2.2

Spike 1.2 Turin

Strelka 2.0

Shredder 9

Glaurung 2.1