Peter Ripota: Wie Computer den besten Zug auswählen

(Quelle: https://www.pm-magazin.de/ - P.M. Computerheft – März 1984) (photo copyright © by http://www.schaakcomputers.nl/) (600 dpi)

Schach mit elektronischen Gegnern

Wie Computer den besten Zug auswählen

Auf den folgenden Seiten kann man selbst nachvollziehen, wie ein Elektronenrechner über die Frage »nachdenkt«, ob er beim Schach mit dem Königsbauern eröffnen soll oder mit dem Damenbauern. Sie werden sehen: Ein menschlicher Schachspieler muß auf diese Frage viel weniger Kombinationen »durchdenken« als der Computer.

Als der fahrende Schausteller Johann Nepomuk Maelzel (einer der Erfinder des Metronoms) in der Mitte des vorigen Jahrhunderts die USA bereiste und dort seinen schachspielenden Türken vorstellte, war unter den Zuschauern in Boston auch der berühmte Schriftsteller Edgar Allan Poe. Poe erkannte (wie vor ihm schon andere), daß der Türke getürkt sein müsse. Tatsächlich stammt der Ausdruck »türken« von den Erfahrungen der Zeitgenossen mit einer Maschine, die gar keine war.

Poe zog in seinem Essay über »Maelzels Schachautomaten« (1836) einige interessante Schlußfolgerungen, von denen eine besonders erwähnenswert ist. Sie ist nämlich ebenso bemerkenswert wie falsch. Poe schreibt:

*Nicht immer bleibt der Schachtürke Sieger. Wäre die Maschine jedoch ein Apparat, der einzig und allein von sich aus funktioniert, so könnte dies nicht der Fall sein – sie würde **jedes** Spiel gewinnen.*

Und Poe liefert auch gleich die Begründung:

*Ist das Prinzip erst einmal entdeckt, nach welchem man eine Maschine dazu bringen kann, Schach zu spielen, so bedarf es bloß einer Erweiterung dieses Prinzips, sie das Spiel auch **gewinnen**, und einer neuerlichen Erweiterung, sie **jedes** Spiel gewinnen zu lassen.*

Schön wär's! Bis jetzt jedenfalls ist es

Peter Ripota: Wie Computer den besten Zug auswählen

(Quelle: https://www.pm-magazin.de/ - P.M. Computerheft – März 1984) (600 dpi)

Die Frage der Fragen: Gibt es ein Prinzip, nach dem man jedes Schachspiel sicher gewinnen kann?

nicht gelungen, jenes allgemeine Prinzip zu finden, das Poe als ganz selbstverständlich voraussetzte. Seit es Computer gibt, bemühen sich ihre Erbauer, Betreuer und Benutzer, der seelenlosen Maschine das königliche Spiel beizubringen. Mit erstaunlichem Erfolg. Immerhin bieten Schach-Computer Großmeistern Remis und besiegen sie gelegentlich. Das erfolgreichste Schachprogramm ist derzeit so stark, daß es 99,9 Prozent aller menschlichen Schachspieler besiegt. Wie das?

Die Idee, einen Computer zum Schachspielen zu bringen, ist so alt wie die Idee, einen Computer zu bauen – und diese Idee stammt bekanntlich von Charles Babbage. Bereits 1864 machte er sich Gedanken darüber, wie er seiner »Analytischen Maschine« das Schachspielen beibringen könnte. Wahrscheinlich hat er's dann doch sein lassen und lieber mit seiner Mitarbeiterin, der Gräfin Ada, ein Spielchen gespielt.

Der erste echte Schachautomat stammt von dem spanischen Erfinder Leonardo Torres y Quevedo. Um 1890 baute er ein (heute noch funktionierendes) Gerät, das ein spezielles Endspielproblem (Turm und König gegen König) lösen sollte. Der Automat glich unseren modernen kleinen Schach-Computern insofern, als er Sensoren für die Position der Figuren und Roboterarme für deren Bewegung besaß. Das Programm (also das Verfahren zum Mattsetzen des schwarzen Königs) wurde durch elektrische Schaltungen realisiert, es war also, in unserer modernen Technologie, »festverdrahtet«.

Auch der Erbauer des ersten elektronischen Rechners, Konrad Zuse, war fasziniert von den Möglichkeiten der Automatisierung des Schachspiels. Er schreibt:

Auf kleinem Raum und mit wenigen Elementen konnte ich beim Schachspiel eine komplizierte Verflechtung von Bedingungen, Fallunterscheidungen und dergleichen studieren. Ich suchte einen Weg, um diese Gesetze mit Hilfe der mathematischen Logik zu formalisieren. Es mußte doch möglich sein, eines Tages mit einer Rechenmaschine den Schachweltmeister zu besiegen.

Zuse glaubte – wie Poe – an ein allgemeines, wenn auch sehr kompliziertes Verfahren, mit Hilfe dessen man jedes Schachspiel gewinnen könne, und er sagte sogar voraus, daß Schachprogramme Zufallsgeneratoren besitzen müßten, damit man sich psychologisch nicht auf sie einstellen und damit leichter besiegen kann.

Das Schachspiel diente als Vorbild für die Entwicklung der »Spieltheorie« (John von Neumann und Oskar Morgenstern). Die Autoren leiteten daraus das sogenannte »Minimax-Theorem« ab, das später tatsächlich Grundlage aller Schachprogramme wurde, wie dieser Bericht noch näher erläutern wird.

Computerschach: Seit 1950 kennt man die Grundstrategien

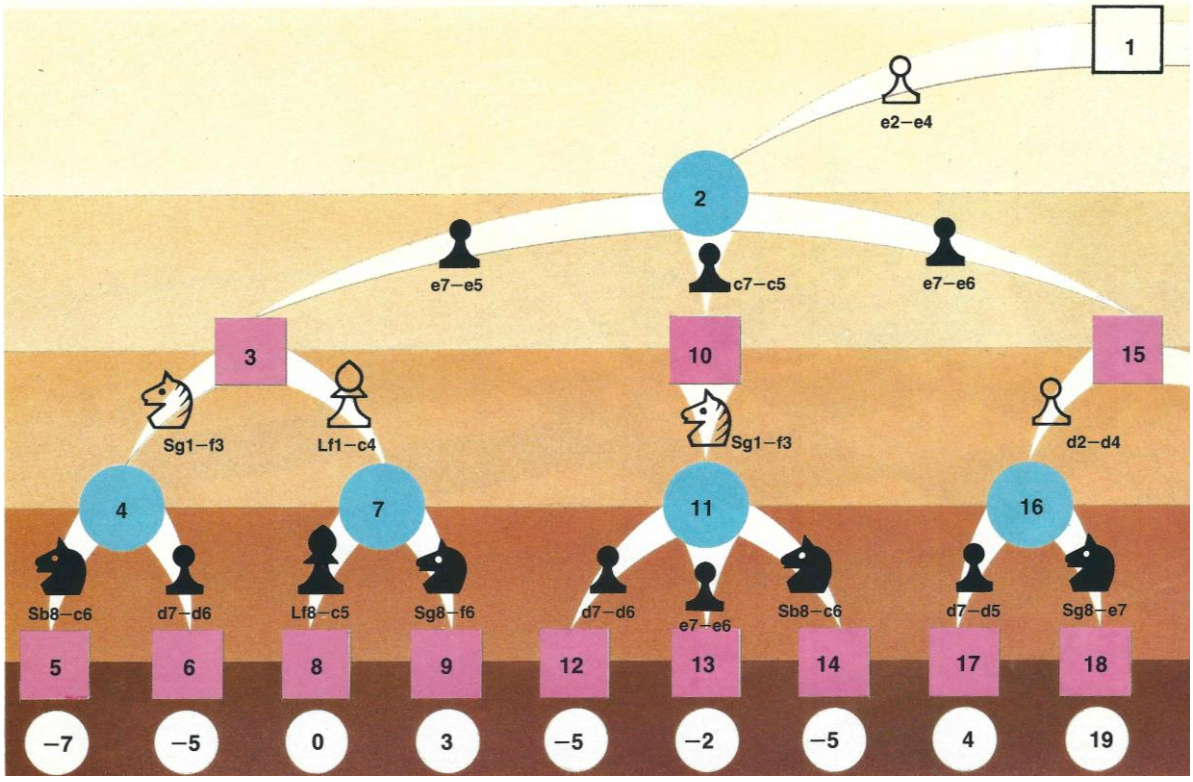
In den fünfziger Jahren polarisierte sich die Einstellung zu den Möglichkeiten. Auf der einen Seite stand der unbekümmerte Optimismus des ehemaligen Weltmeisters Michail Botwinnik, der 1955 sagte:

Im Lauf der nächsten 15 Jahre werden Computer fähig sein, jeden Großmeister zu schlagen.

Den pessimistischen Pol verkörperte Norbert Wiener, Erfinder der Kybernetik, als er meinte:

Es ist ein hoffnungsloses Unterfangen, eine Maschine dahin zu bringen, perfektes Schach zu spielen.

1950 war ein entscheidendes Jahr in



Peter Ripota: Wie Computer den besten Zug auswählen

(Quelle: https://www.pm-magazin.de/ - P.M. Computerheft – März 1984) (photo copyright © by http://www.schaakcomputers.nl/) (600 dpi)

der Geschichte des Computerschachs. Damals formulierte der Entwickler der Informationstheorie und Entdecker des »Bit«, Claude Shannon, grundlegende Überlegungen zur »Programmierung eines Rechners zum Schachspielen«, wie der Titel seines Vortrags und seines Artikels in der Zeitschrift Scientific American hieß. Es war die erste, gründlichste und brauchbarste Untersuchung ihrer Art, die meisten darin formulierten Ideen haben heute noch Gültigkeit.

Die ***A-Strategie,*** auch Methode der »Rohen Gewalt« (brute force) genannt, erzeugt systematisch alle legalen Züge und damit alle möglichen Stellungen. Vorteil: Es wird nichts übersehen. Nachteil: Der Aufwand zur Generierung, Speicherung und Bewertung der Positionen ist ungeheuer. Bis jetzt waren reine A-Programme die erfolgreichsten.

Bei der ***B-Strategie*** (heuristische Programmierung) ist im Gegensatz zur »rohen Gewalt« der A-Strategie die Anzahl der vorausberechneten Züge variabel. Sie hängt ab von theoretischen und praktischen Überlegungen, die unmittelbar mit dem Schachspiel zu tun haben. Hier wird also in das Programm Schachwissen eingetrichtert, auf Grund dessen eine Vorauswahl und Reduzierung von Zügen durchgeführt werden kann.

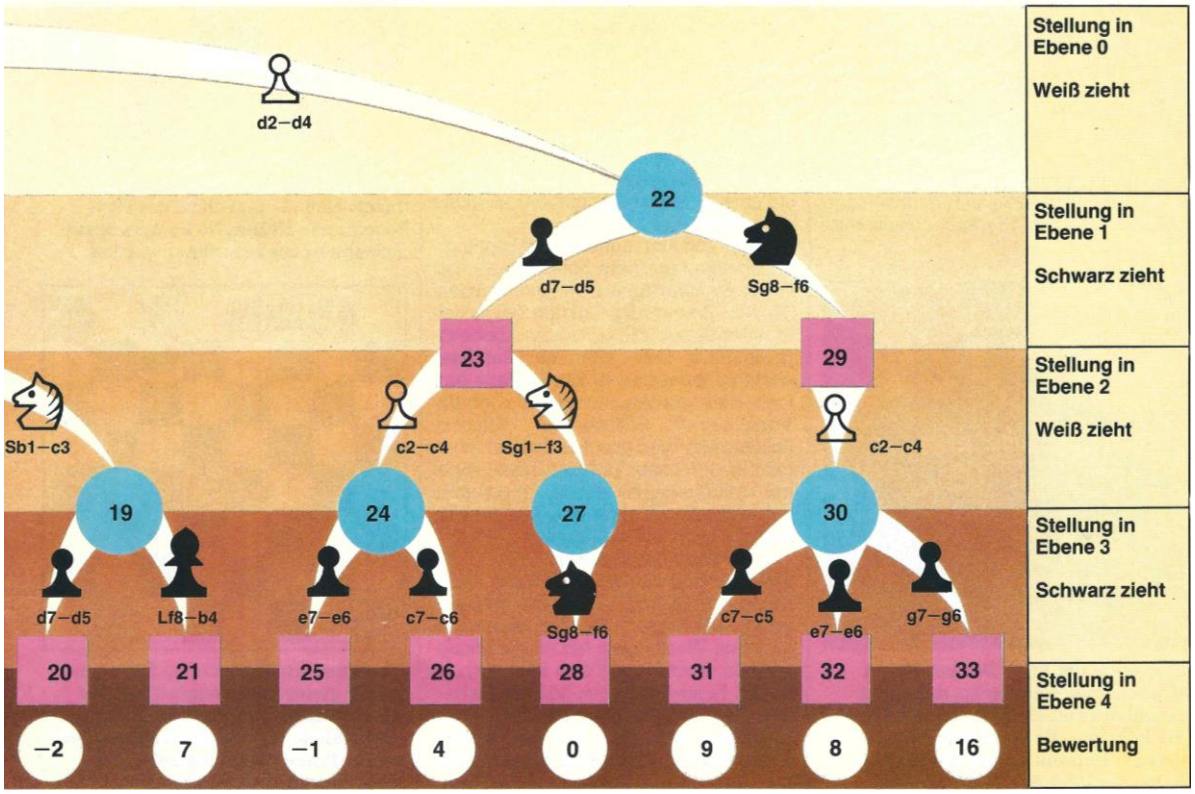
Die ***C-Strategie*** (Lernende Programme, Künstliche Intelligenz) ist der ehrgeizige Versuch, menschliche intellektuelle Fähigkeiten über das Schachspiel auf einem Computer nachzubilden. Versuche in dieser Richtung (Botwinnik in der Sowjetunion, Newell, Shaw und Simon in den Vereinigten Staaten) sind bisher gescheitert. Die Unterschiede zwischen dem »Denken« des Computers und den Erkenntnis- und Lernvorgängen im Menschen sind offenbar zu groß. Eine erfolgreiche Realisierung dieser Strategie wird vermutlich erst möglich sein, wenn völlig andere Rechnerkonzepte verfügbar sind (Assoziativspeicher, Parallel-Prozessoren).

Shannon selbst behauptete nicht, Computerschach hätte irgendwelchen praktischen Nutzen. Aber er meinte, daß eine befriedigende Lösung dieses Problems auch zu Forschritten auf anderen Gebieten des automatischen Problemlösens führen könnte. Besonders dachte er an die Möglichkeit, Maschinen zu bauen (das heißt, Programme zu entwickeln) zum automatischen Entwurf elektronischer Schaltkreise, zur Sprachübersetzung, zur Unterstützung strategischer Entscheidungen bei militärischen Operationen, zum Komponieren und zum automatischen Beweisen mathematischer Theoreme. Das sind alles Forschungsgebiete der »Künstlichen Intelligenz« (Artficial Intelligence, abgekürzt AI), die zum Teil auch schon befriedigend gelöst wurden – aber nicht mit den Methoden, nach denen ein Computer Schach spielt. Was die AI-Leute selbst am meisten bedauern! Sie haben sich bis jetzt vergeblich bemüht, brauchbare Schachprogramme zu schreiben, die ein gewisses Mindestmaß an »Intelligenz« und Lernfähigkeit besitzen. Ihre Hauptvertreter, Newell, Shaw und Simon, formulierten ihre Ansichten so:

Computerschach: Bis heute dominiert die »Rohe Gewalt«

Wir möchten nicht nur, daß unser Programm gute Züge macht. Wir möchten, daß es das auch aus den richtigen Gründen tut. Könnte jemand eine erfolgreiche Schachmaschine entwickeln, dann wäre er zum Kern der geistigen Aktivität des Menschen vorgedrungen.

In dieser Grafik zeigt die oberste (hellste) Ebene die beiden Eröffnungszüge, über die der »weiße« Computer nachdenkt. Ebenen darunter: die nächsten drei Halbzüge. In runden Feldern sieht man mögliche Stellungen, die durch Züge weißer Figuren entstehen. Quadratische Felder zeigen Konsequenzen schwarzer Züge. Alle Züge werden vom »weißen« Computer vorausgesehen und bewertet. Die Zahlen geben die Reihenfolge der Zug-Erzeugung an. Ganz unten: die abschließenden Bewertungen.



Peter Ripota: Wie Computer den besten Zug auswählen

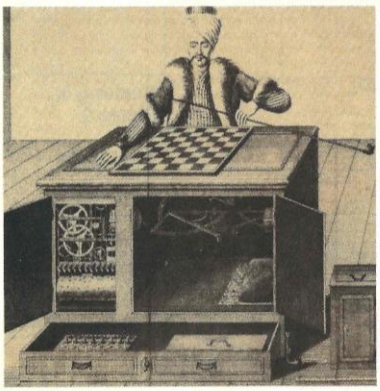
(Quelle: https://www.pm-magazin.de/ - P.M. Computerheft – März 1984) (photo copyright © by http://www.schaakcomputers.nl/) (600 dpi)

Wie weiß der Computer, wo die einzelnen Figuren stehen und wen sie schlagen können?

Hier unterbrechen wir unseren Spaziergang durch die Geschichte des Computerschachs und stellen uns die Frage: Wie spielt ein Computer Schach?

Bevor ein Schach-Computer Figuren in seinem Inneren hin- und herschiebt, muß er sie irgendwie erkennen und ihre Positionen speichern. Mit anderen Worten, der Computer braucht eine ***Repräsentation des Schachbretts*** und seiner Figuren. Am elegantesten geschieht dies durch sogenannte Bit-Karten. Ein Bit ist die kleinste Informationseinheit. Es kann den Wert 1 oder 0 annehmen – und das kann man in diesem Fall mit »vorhanden« und »nicht vorhanden« deuten. Das Schachfeld wird sozusagen in viele Einzelfelder aufgespalten. Jedes dieser Einzelfelder – jede Bitkarte – enthält nur einen bestimmten Figurentyp. So gibt es eine Bitkarte für die weißen Bauern, eine für die schwarzen Bauern, eine für die weißen Türme… usw. Auf dieser Karte der Größe 8x8 = 64 Felder gibt es nur Einsen oder Nullen, je nachdem, ob die betrachtete Figur auf dem Feld steht oder nicht. Zusätzlich dazu werden auch die Grundzüge der einzelnen Figuren in Bitkarten gespeichert.

Der Vorteil dieser Darstellung – eine Feldebene = ein Wort mit 64 Stellen – liegt darin, daß schachspezifische Fragen durch einfache (und schnelle) logische Operationen beantwortet werden können. Will das Programm beispielsweise wissen, welche Felder frei sind, dann nimmt es einfach die Oder-Verknüpfung sämtlicher Bitkarten – das sind alle irgendwie belegten Felder. Anschließend wird diese Karte negiert, und schon hat man alle leeren Felder. Oder das Programm möchte wissen, welche gegnerischen Figuren durch den eigenen Turm angegriffen werden. Lösung: Nimm die Bit-Karte des eigenen Turms und mache eine UND-Verknüpfung mit den geODERten Bitkarten aller gegnerischen Figuren (Verzeihen Sie die seltsame Wortbildung). Allerdings können noch eigene Figuren die Schlaglinie blockieren. Das muß eigens untersucht werden. Doch das Grundprinzip wird klar.



Sensation um 1790: Der berühmte schachspielende Türke, der sogar Napoleon und Kaiserin Maria Theresia besiegte. Heute wissen wir: In seinem Inneren waren die besten Schachspieler der Zeit versteckt.

Das Programm baut sich einen sogenannten Spielbaum auf. Ausgehend von der gegenwärtigen Stellung (Position) werden alle möglichen Züge und Gegenzüge bis zu einer bestimmten Tiefe erzeugt. Das geschieht durch den ***Zuggenerator,*** einem der wichtigsten Teile eines jeden Schachprogramms. Der Zuggenerator erzeugt nach einem bestimmten Schema alle möglichen Züge. Von diesen müssen jetzt alle erlaubten und danach alle sinnvollen Züge ausgesondert werden. (Noch besser wäre es, unerlaubte und sinnlose Züge würden gar nicht erst erzeugt.) Erlaubt ist ein Zug dann, wenn durch ihn nicht eine eigene Figur geschlagen oder übersprungen wird (Ausnahme: Der Springer darf alle Figuren überspringen), und wenn der eigene König nicht ins Schach gerät.

Wann ein Zug sinnvoll ist, das zu entscheiden ist schon viel schwieriger. Sicherlich ist ein Zug sinnlos, bei dem eine eigene Figur ohne sichtbare Vorteile geopfert wird. Bereits in diesem Stadium müssen daher heuristische Regeln (also Regeln, die aus der Erfahrung stammen) herhalten, um die Zugfülle einzudämmen.

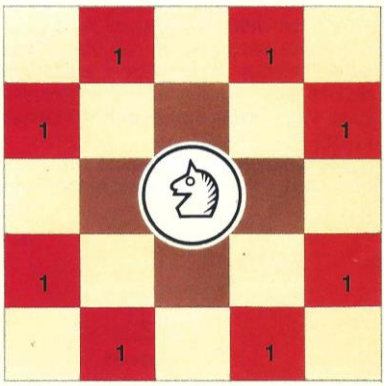
Jeder Zug führt nun zu einer bestimmten Stellung auf dem Schachbrett, und diese Stellung muß bewertet werden. An der Bewertungsfunktion zeigt sich unter anderem die Güte eines Schachprogramms. Das materielle Gleichgewicht zu bewerten ist noch relativ einfach. Hier werden auch heute noch die Vorschläge Shannon's realisiert. Schwieriger wird es schon bei der Beurteilung der Lage. Um den Leser nicht mit Allgemeinplätzen zu langweilen, konstruieren wir eine einfache Bewertungsfunktion. Sie sieht so aus:

1. ***Gewicht der Figuren.*** Bauer = 10, Springer und Läufer = 30, Turm = 50, Dame = 90, König = 1000. Die gegnerischen (schwarzen) Figuren haben die gleichen Werte, aber mit einem Minuszeichen davor. Die Werte aller vorhandenen Figuren werden addiert. Haben die beiden Spieler die gleichen Figuren, dann ist dieser Teil der Bewertungsfunktion gleich Null, braucht also nicht beachtet zu werden.

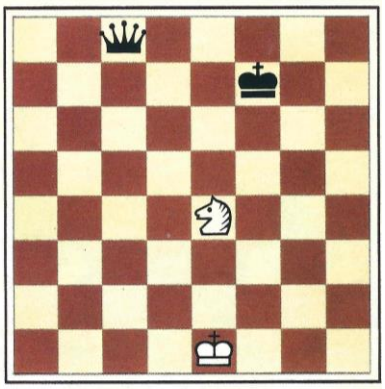
2. ***Beweglichkeit der Figuren.*** Wir zählen einfach für jede Figur die Anzahl der Felder, auf die die Figur in der gegenwärtigen Lage gestellt werden kann. Beim Gegner geht diese Zahl, wie üblich, mit einem Minuszeichen in die Rechnung ein.

3. ***Königssicherheit.*** Sie hängt vom Entwicklungsstand der Partie ab. Für den Anfang (die Eröffnung) genügt als Maß für die Sicherheit des Königs die Anzahl der Züge bis zur vollendeten Rochade. Sie ist nicht größer als 4. Damit diese Zahl in der Gesamtbewertung nicht untergeht, multiplizieren wir sie mit 3. Sie hat also ein anderes mathematisches »Gewicht« als die Figurenbeweglichkeit.

4. ***Zentrumskontrolle.*** Auch hier machen wir uns die Sache leicht. Wir betrachten die vier Zentrumsfelder d4, d5, e4 und e5, und zählen die Zahl der angegriffenen Felder. Auch diese Zahl liegt um 4, so daß wir sie ebenfalls mit 3 multiplizieren. Mehrfach angegriffene Felder



»Bit-Karte« für einen Springer. Alle Felder, die er erreichen kann, sind durch eine »1« markiert, der Rest besteht aus Nullen. So können Springerzüge leicht berechnet werden.



Kann der weiße Springer Dame und König gleichzeitig angreifen? Der menschliche Spieler sieht das auf einen Blick, der Computer beantwortet die Frage durch logische Verknüpfung mehrerer Bitkarten.

Peter Ripota: Wie Computer den besten Zug auswählen

(Quelle: https://www.pm-magazin.de/ - P.M. Computerheft – März 1984) (photo copyright © by http://www.schaakcomputers.nl/) (600 dpi)

zählen auch dementsprechend mehr.

Jetzt bauen wir unseren ***Spielbaum*** auf. Als Ausgangslage wählen wir den Beginn einer Partie. Alle Figuren sind also noch auf ihren Ruheplätzen. Anstelle der 20 möglichen Züge für Weiß sehen wir uns aber nur 2 Züge mit ihren Folgen an. Im Bild auf den Seiten 42/43 sind einige der Züge und Gegenzüge (bei weitem nicht alle!) bis zur Tiefe von vier Halbzügen (zwei für Weiß, zwei für Schwarz) aufgezeichnet. Die Reihenfolge, in der der Spielbaum aufgebaut wird, entspricht der Numerierung. Man spricht bei diesem Verfahren von einer ***Tiefensuche,*** weil das Programm zuerst in die Tiefe geht. Bei einer ***Breitensuche*** (die ebenfalls gern angewandt wird) werden zuerst alle möglichen Züge auf Ebene 1 erzeugt, dann alle möglichen Gegenzüge, usw.

Sind nun alle Endknoten (= Endstellungen) erzeugt und im Computergedächtnis gespeichert, dann werden diese Endstellungen bewertet. Das heißt, die Bewertungsfunktion wird auf die entsprechende Verteilung der Figuren angewendet. Die Bewertungszahlen stehen unter den Endknoten. Das materielle Gleichgewicht muß im Eröffnungsstadium nicht mitgezählt werden, da die beiden Spieler noch alle Figuren haben. Wie man zu den Zahlen kommt, zeige ich an einem Beispiel, und zwar am Spielstand, der dem Knoten »6« entspricht. Die Partie nahm bis dahin folgenden Verlauf:

1. e2–e4 e7–e5
2. Sg1–f3 d7–d6

Wer sich nun die Mühe macht, die Stellung aufzubauen und zu zählen, der wird erkennen, daß Weiß mit seinen Figuren insgesamt 27 Felder belegen kann, Schwarz dagegen 32. Weiß kann in zwei Zügen rochieren, Schwarz in drei Zügen. Weiß greift mit seinen Figuren 3 Zentrumsfelder an, Schwarz deren 2. Daraus ergibt sich:

(1) Figurengewicht = 0 (gleichviele Figuren auf beiden Seiten).
(2) Figurenbeweglichkeit = 27–32 = –5
(3) Königssicherheit = 3x(2–3) = –3
(4) Zentrumskontrolle = 3x(3–2) = 3

Addiert man alle Zahlen, dann ergibt sich als Bewertung dieser Zugfolge (und damit dieser Endstellung) die Zahl –5 für den Knoten mit der Nummer 6. Genauso verfährt das Programm mit den anderen Endknoten.

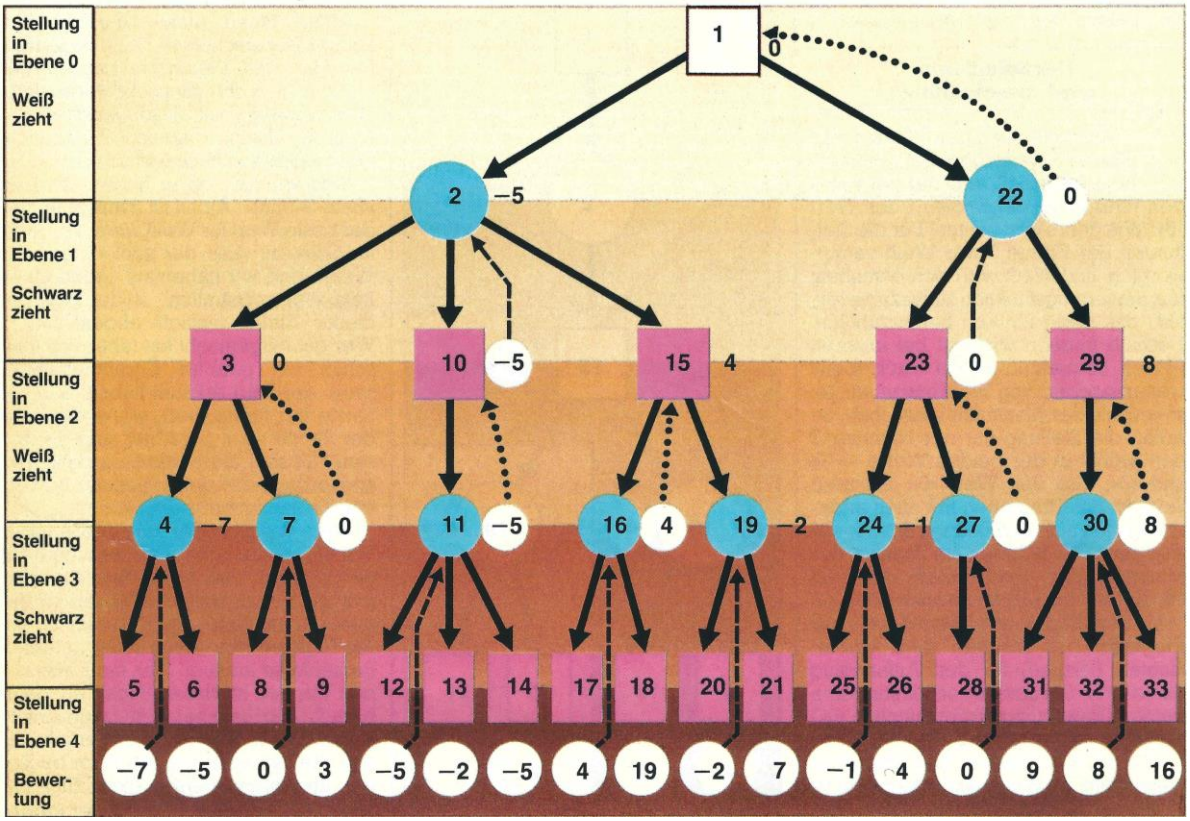
Spielverläufe (also Zugfolgen) und Bewertungen werden gespeichert, und danach – ja, was nun? Jetzt muß das Programm einen Zug auswählen – nicht irgendeinen, sondern den besten (auf Grund seiner Bewertungen). Ein naiver Betrachter würde meinen, das Programm wählte jenen Zug, der letzten Endes zu dem Knoten mit der höchsten Bewertung führt. Das wäre in unserem Beispiel die Endstellung mit der Nummer 18.

Das Minimax-Prinzip bestimmt den Zug

Aber so einfach ist die Sache nicht. Schwarz hat nämlich auch noch ein Wort mitzureden, und zwar bestimmt es jeden zweiten Zug. Und Schwarz wird natürlich versuchen, den Gewinn von Weiß möglichst wieder zunichte zu machen. Oder mathematisch ausgedrückt: Weiß bemüht sich darum, die Bewertungsfunktion jeweils zu einem Maximum zu machen; Schwarz bemüht sich darum, diese (auf Weiß abgestimmte) Bewertungsfunktion auf ein Minimum zu drücken. Schwarz minimiert das Maximum von Weiß. Und darum heißt diese Strategie (die den meisten Spielen zugrundeliegt und von John von Neumann entdeckt wurde) auch die ***Minimax-Strategie.***

Der Vorteil dieser Strategie liegt darin, daß man sie einem Computer sehr einfach beibringen kann. Der menschliche »Zuschauer« hat mehr Schwierig-

Zugbewertung mit dem Beispiel von Seite 42/43. Weiß geht nach der größten Zahl (Maximum), Schwarz nach der kleinsten (Minimum). Die Entscheidung fällt schließlich für den Zug d2-d4, da dort das Programm den größten Wert errechnet (0 statt –5).



Peter Ripota: Wie Computer den besten Zug auswählen

(Quelle: https://www.pm-magazin.de/ - P.M. Computerheft – März 1984) (photo copyright © by http://www.schaakcomputers.nl/) (600 dpi)

Wie der Computer sinnlose Züge ausschaltet und damit wertvolle Rechenzeit dazugewinnt

keiten, sie nachzuvollziehen. Wir werden zeigen, daß das Beste nicht immer das Beste ist.

Fangen wir ganz unten an (siehe Bild auf Seite 45), bei den Endstellungen. Sie wurden durch Züge von ***Schwarz*** erreicht. Schwarz wird sich aber bemühen, den Wert dieser Stellungen möglichst zu drücken. Mit anderen Worten: Wenn's nach Schwarz geht (und auf dieser Ebene geht es nach Schwarz!), dann werden nur die Stellungen erreicht, die ein ***Minimum*** der Bewertungsfunktion ergeben. Immer dort, wo also mehrere Möglichkeiten für Schwarz bestehen, wird der schwarze Spieler diejenige Möglichkeit auswählen, die ihm den kleinsten Nachteil bringt.

Konkret gesagt: Schwarz hat die Wahl zwischen Stellung 5 und 6. Und Schwarz wird die Stellung 5 auswählen, weil hier sein Kontrahent schlechter abschneidet als bei Stellung 6 (Bewertung von 5 ist kleiner als die Bewertung von 6). Und nur dieser Wert – im Beispiel »–7« – ist für die Entscheidung bedeutungsvoll!

Der Spielbaum wird »beschnitten«

Das Minimum der Ebene 4 wird jetzt nach Ebene 3 weitergegeben. In unserem Beispiel erhält also der Knoten 4 den Wert »–7«, der Knoten 7 den Wert »0«. Wie geht's jetzt weiter? Für die Stellungen der Ebene 3 war Weiß verantwortlich, und Weiß wird sich bemühen, die Bewertungsfunktion zu maximieren, also das Beste für sich herauszuholen. Deshalb müssen wir jetzt bei unseren Prozeß, Bewertungszahlen »nach oben« weiterzugeben, von Weiß ausgehen, also jeweils das Maximum aussuchen. So erhält die Stellung mit der Nummer 3 den größeren der beiden Werte »–7« und »0«, also den Wert »0«. Genauso verfährt das Programm mit allen anderen Knoten auf allen Ebenen. Immer abwechseln: Wo Schwarz am Zuge ist, das Minimum nach oben geben, wo Weiß am Zug ist, das Maximum auswählen.

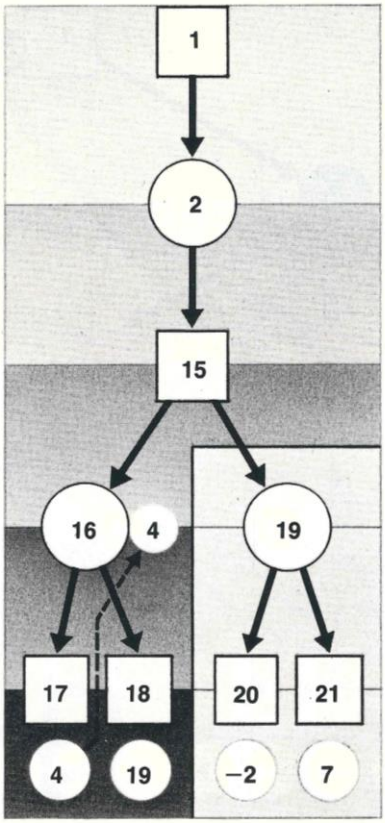
Letzten Endes entscheidet sich das Programm (in unserem stark vereinfachten Beispiel) für den Anfangszug d2–d4, da sich daraus noch immer eine bessere Bewertung ergibt (nämlich »0«) als für den Alternativzug e2–e4.

Wie es nach diesem Zug allerdings weitergeht, weiß niemand. Denn jetzt ist ja wieder Schwarz an der Reihe, und wenn Schwarz genauso denkt wie Weiß (und wie der Computer), dann wird er c7–c5 wählen, weil er so seinem Gegner die größten Schmerzen bereitet.

Alles etwas mühsam? Nicht für einen Rechner, denn solche Entscheidungen sind sein tägliches Brot. Und Vorstellungsvermögen hat er ohnedies nicht. Also stört ihn auch nicht (im Gegensatz zum Menschen), daß dieses Verfahren recht unanschaulich ist.

Nun haben sich die Programmierer noch einen hübschen Trick ausgedacht, wie die zu untersuchende Zugzahl eingeschränkt werden kann. Das Verfahren nennt sich ***Alpha-Beta-Algorithmus,*** und wir zeigen wieder an einem Beispiel, wie es funktioniert. Dazu nehmen wir die Zugfolge aus dem Bild auf dieser Seite. Zuerst untersuchen wir den linken Zweig des Spielbaums, das heißt, wir entscheiden uns zwischen den Stellungen 17 und 18. Da hier Schwarz am Zug war, wählen wir das Minimum der

Der Alpha-Beta-Algorithmus kann die Zahl der Züge, die untersucht werden müssen, erheblich beschränken. Im Beispiel werden die Züge des Knotens 19 gar nicht mehr untersucht, da sie ganz unten zu dem Minimum –2 führen. Das Minimum links ist aber +4, also größer und damit günstiger für Weiß. Der Spielbaum kann hier beschnitten werden, was enorme Ersparnisse an Zeit mit sich bringt.



beiden Bewertungen, die Zahl 4. Knoten 16 erhält also den Wert »4«. Jetzt müssen wir das gleiche für Knoten 19 machen.

Aber vorher noch eine kleine Überlegung. Es gibt ja nur zwei Möglichkeiten: Die Stellung 19 ist entweder besser oder aber schlechter als die Stellung 16. Wenn nun Stellung 19 schlechter sein sollte als 16, dann interessiert sie uns nicht, denn auf dieser Ebene ist ja wieder Weiß am Zug, und Weiß maximiert, wählt nur die größte Zahl aus.

Kann man schon eine Vorentscheidung über den Wert von 19 treffen? Man kann! Denn die Stellung 19 kann nicht besser sein als der schlechteste Wert aller von ihr abgeleiteten Stellungen. Grund: Die abgeleiteten Stellungen werden von Schwarz bestimmt – und Schwarz sucht sich diejenige mit dem schlechtesten Wert für Weiß aus. Die erste »Tochter«-Stellung von 19 hat den Wert –2 (Stellung 20). Besser kann 19 also nicht werden. Also ist 19 auf jeden Fall schlechter als 16, und damit für die weitere Bewertung uninteressant.

Darum brauchen die Töchter (oder Söhne, wie man's nimmt) von 19 gar nicht weiter untersucht werden – im allgemeinen ein ungeheurer Zeitgewinn, auch wenn man das in unserem Beispiel nicht so recht sieht. Aber wir haben ja auch nur ein bis drei Töchter untersucht. In Wirklichkeit sind es bis zu 40! Und alle haben sie wieder Töchter und Söhne, und das geht ganz ungeheuer in die Zeit – und in den Speicher.

Wenn Ihnen diese Überlegungen kompliziert erscheinen, dann bedenken Sie, daß das Verfahren computergerecht ist, nicht menschenfreundlich. Dem Computer ist diese Denkweise – Maxima, Minima, ja/nein – natürlich. Darum geht's einfach und schnell.

Was wir hier getan haben, war ein »Beta-Schnitt«. Alpha ist nämlich immer der beste Wert für Weiß, Beta der beste für Schwarz (also der schlechteste für Weiß). Und wir haben auf Grund eines Beta-Wertes (nämlich –2) den Baum an dieser Stelle einfach abgeschnitten. Wer gut mitgemacht hat (aber das muß schon ein geübter Computer-Freak sein), der wird erkannt haben, daß die Sache nur funktioniert, wenn die Züge der Größe nach geordnet sind. In diesem Fall sind die Endknoten zufällig so angeordnet, daß der schlechter bewertete Knoten immer zuerst kommt.

Wäre das nicht der Fall, würde unser Verfahren versagen. Also steht und fällt die Effizienz des Alpha-Beta-Algorithmus mit einem Sortierverfahren für die generierten Züge. Noch besser ist es, die Züge werden gleich in der richtigen Reihenfolge erzeugt. Man fängt also immer mit den stärksten Zügen an, dann braucht man nachher nicht lange zu sortieren. Starke Züge sind sicherlich solche, die eine gegnerische Figur bedrohen, und darum werden diese Züge immer zuerst erzeugt.

Peter Ripota

Peter Ripota: Wie Computer den besten Zug auswählen

(Quelle: https://www.pm-magazin.de/ - P.M. Computerheft – März 1984) (photo copyright © by http://www.schaakcomputers.nl/) (600 dpi)

Kleine Geschichte des Computerschachs

1836 *Johann Nepomuk Maelzel* bereist die USA und führt in Boston seinen schachspielenden Türken vor. *Edgar Allan Poe* erkennt, daß der Türke getürkt ist und schreibt darüber ein Essay mit interessanten Gedanken zur Automatisierung des Schachspiels.

1864 *Charles Babbage* überlegt, wie seine Analytische Maschine Schach spielen könnte.

1890 *Leonardo Torres y Quevedo* konstruiert den ersten echten Schachautomaten, der ein bestimmtes Turmendspiel in 63 Zügen löst. Der Automat war gleichzeitig ein Roboter, da er Sensoren für die Figuren und Greifarme für deren Fortbewegung besaß.

1945 *Konrad Zuse* schreibt einige Schachalgorithmen in seinem »Plankalkül«.

1949/50 *Claude Shannon* stellt in einem bahnbrechenden Vortrag/Artikel drei Methoden der Schachprogrammierung vor. Seine Ideen haben heute noch Gültigkeit.

1952 *Alan Turing* von der Universität Manchester entwickelt einen Schachalgorithmus, der von Hand simuliert wird. Sein Programm »Turochamp« verliert gegen einen 26jährigen Studenten. Turochamp rechnet in ruhigen Stellungen zwei Halbzüge voraus.

1955 Erste Schachprogramme in den USA *(Bernstein, Newell-Shaw-Simon).*

1965 Erstes deutsches Schachprogramm von *Fischer* und *Schneider.*

1966 Ein sowjetisches Schachprogramm *(Adelson-Belski* und *Alasarow)* schlägt im Fernschach ein amerikanisches Schachprogramm *(McCarthy, Kotok).*

1967 Ein Schachprogramm (MacHack VI von *Richard Greenblatt)* wird Ehrenmitglied zweier amerikanischer Schachgesellschaften.

1970 Erstes Computerschachturnier der ACM (Association for Computing Machinery) in New York. Sieger: Chess 3.0 von *Larry Atkin* und *David Slate,* Northwestern University, Illinois.

1974 Erste Computerschach-Weltmeisterschaft in Stockholm. Weltmeister: das sowjetische Programm *Kaissa.*

1976 Computerschach-Weltmeister *Chess* 4.5 gewinnt ein Turnier gegen 5 Spieler der Klasse B.

1977 Die ersten kommerziell verfügbaren Schach-Computer (Chess Challenger) und Schachprogramme (Boris, Sargon) kommen auf den Markt. Preis: 150 bis 200 Dollar.

1978 *Chess* 4.7 siegt in einer Partie gegen den Internationalen Meister *David Levy.* Levy gewinnt aber trotzdem seine Wette, kein Computer werde ihn innerhalb von 10 Jahren in einem Turnier besiegen.

1978 Die Computerschach-Weltmeisterschaft geht auf *Belle* über (entwickelt von *Ken Thompson* und *Joe Condon).*

1978 Bei der neunten ACM-Computerschach-Weltmeisterschaft in Washington belegt das Mikrocomputer-Schachprogramm *Sargon II (Dan* und *Kathe Spracklen)* den dritten Platz (gemeinsam mit Chaos und Blitz).

1983 *Belle* erreicht über 2300 Elo-Punkte und ist damit besser als 99,9 Prozent aller menschlichen Schachspieler.

1983 Bei der Computerschach-Weltmeisterschaft in New York schneidet das Mikrocomputer-Schachprogramm *Mephisto* punktgleich mit dem bisherigen Weltmeister *Belle* ab.

1988 (oder auch früher): Ein Schachprogramm wird internationaler Großmeister.

1992 (oder auch früher): Schachprogramme schlagen jeden Menschen, auch den amtierenden Weltmeister.

Peter Ripota: Wie Computer den besten Zug auswählen

(Quelle: https://www.pm-magazin.de/ - P.M. Computerheft – März 1984) (600 dpi)

Kleine Geschichte des Computerschachs…

Fidelity Chess Challenger (1) - Release: März 1977

1977: Die ersten kommerziell verfügbaren Schachcomputer (Chess Challenger) und Schachprogramme (Boris, Sargon) kommen auf den Markt. Preis: 150 bis 200 Dollar.

Schach mit elektronischen Gegnern

Wie Computer den besten Zug auswählen

Auf den folgenden Seiten kann man selbst nachvollziehen, wie ein Elektronenrechner über die Frage »nachdenkt«, ob er beim Schach mit dem Königsbauern eröffnen soll oder mit dem Damenbauern. Sie werden sehen: Ein menschlicher Schachspieler muß auf diese Frage viel weniger Kombinationen »durchdenken« als der Computer.

Als der fahrende Schausteller Johann Nepomuk Maelzel (einer der Erfinder des Metronoms) in der Mitte des vorigen Jahrhunderts die USA bereiste und dort seinen schachspielenden Türken vorstellte, war unter den Zuschauern in Boston auch der berühmte Schriftsteller Edgar Allan Poe. Poe erkannte (wie vor ihm schon andere), daß der Türke getürkt sein müsse. Tatsächlich stammt der Ausdruck »türken« von den Erfahrungen der Zeitgenossen mit einer Maschine, die gar keine war.

Poe zog in seinem Essay über »Maelzels Schachautomaten« (1836) einige interessante Schlußfolgerungen, von denen eine besonders erwähnenswert ist. Sie ist nämlich ebenso bemerkenswert wie falsch. Poe schreibt: *Nicht immer bleibt der Schachtürke Sieger. Wäre die Maschine jedoch ein Apparat, der einzig und allein von sich aus funktioniert, so könnte dies nicht der Fall sein – sie würde* ***jedes*** *Spiel gewinnen.*

Und Poe liefert auch gleich die Begründung: *Ist das Prinzip erst einmal entdeckt, nach welchem man eine Maschine dazu bringen kann, Schach zu spielen, so bedarf es bloß einer Erweiterung dieses Prinzips, sie das Spiel auch* ***gewinnen****, und einer neuerlichen Erweiterung, sie* ***jedes*** *Spiel gewinnen zu lassen.*

Schön wär's! Bis jetzt jedenfalls ist es

Computerheft 41

First Published on March 16, 2019 – Size: 34.051 KB