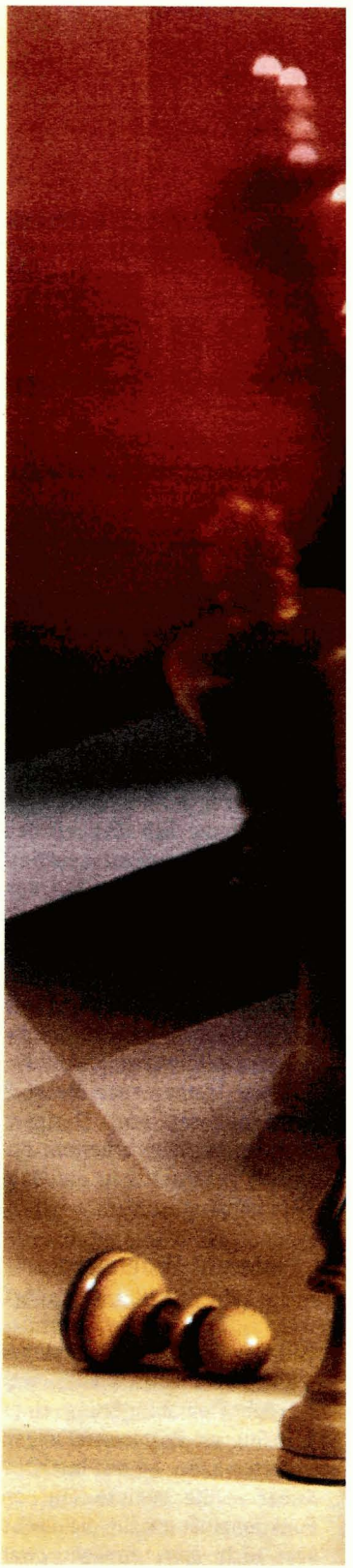
Der PC lernt Schach

Das Spiel der Könige

Dr. Chrilly Donninger, Dr. Klaus Manhart **• Ein Schachprogramm zu schreiben, ist nicht schwer, nicht mal eines, das gegen Gott und die Welt gewinnt. KISS – Keep It Simple and Stupid – lautet eine auf Sieg programmierte Devise.**



Als „Probierstein des Gehirns" hatte schon Goethe jenes herausragende Verstandesspiel bezeichnet, das, indischen Ursprungs, in der zweiten Hälfte des sechzehnten Jahrhunderts aus Persien nach Europa kam. „Shah mat" – wörtlich übersetzt „Der König ist tot" – ist das Ziel des Spiels und erklärt, warum Schach von jeher so beliebt ist: ein Königsmord – welcher Bettler hätte daran keinen Spaß.

Zu den Ärmsten zählen die Schach-Großmeister heute nicht mehr. Und ihre härtesten Gegner, so scheint es, sind keine Könige mehr, sondern Maschinen. Als Pionier der Schachprogrammierung gilt neben dem US-Amerikaner Claude Shannon (1916) der Engländer Alan Turing (1912 – 1954). Er formulierte jenes nach ihm benannte Kriterium, wann eine Maschine „intelligent" sei. Und da er Schach als Spiel erachtete, welches ein besonders hohes Maß an Intelligenz voraussetzt, schuf er mangels geeigneter Hardware einen Papier-Computer. Die erste im Jahre 1952 ausgetragene Partie gewann – wie nicht anders zu erwarten – ein Mensch. Immerhin benötigte der Hobbyspieler Allick Glennie 22 Züge bis zum Matt. Der Niederlage trotzend, begann Turing, ein richtiges Schachprogramm zu schreiben, wurde damit aber nie fertig. Im Juni 1954 starb er unter tragischen Umständen.

Doch warum interessiert Schach eigentlich so viele Computertheoretiker? Die Probleme sind überschaubar, die Figuren, Züge und Felder begrenzt, die Regeln klar definiert und Erfolg oder Mißerfolg eindeutig feststellbar. Andererseits besitzt das Spiel eine ungeheure Komplexität, die niemand ganz durchschaut. Aus der Sicht eines Programmierers ist Schach weder zu einfach noch zu komplex. Wäre es zu einfach, hätte sicher schon jemand ein Programm geschrieben, das den Schach-Weltmeister schlägt (Kasten „Deep Blue"). Und wäre es zu komplex, würde es die modernen Schachprogramme, gegen die 98 Prozent aller Schachspieler keine Chance haben, gar nicht geben.

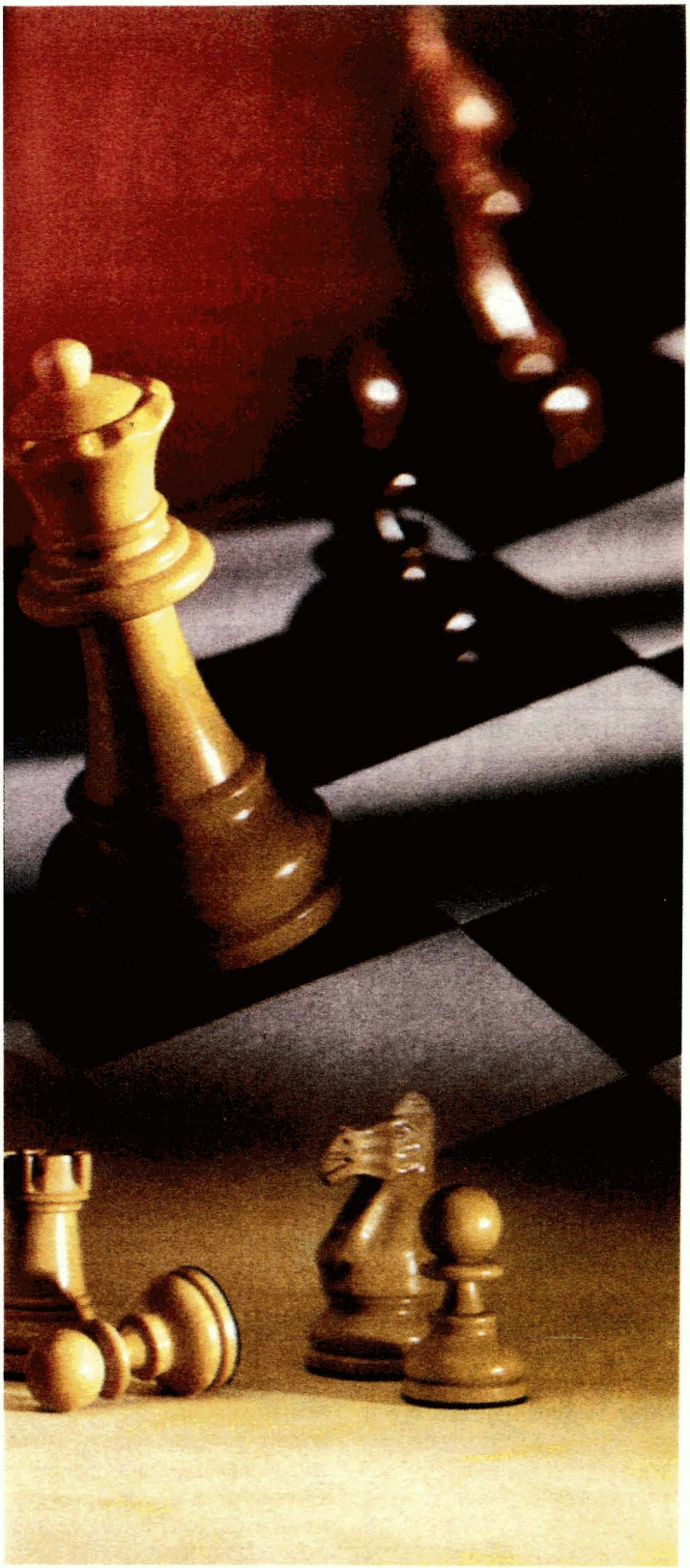
Im Prinzip kann jeder ein Schachprogramm schreiben, der grundlegende Programmier- und Schachkenntnisse hat. Als Programmierer vollkommen ungeeignet ist eigentlich nur eine einzige Spezies Mensch: Schach-Groß- oder Weltmeister. Das bekannteste Beispiel dafür ist Ex-Weltmeister Botwinnik, der fast 30 Jahre lang ohne nennenswerte Resultate an seinem Wunderkind gebastelt hat. Umgekehrt sind die „Großmeister" der Schach-Programmierung in der Regel nur mittelmäßige Schachspieler.

Dreifaltigkeit und Repräsentanz

Ein Schachprogramm besteht im Kern aus drei weitgehend unabhängigen Komponenten: dem Zuggenerator, der Suche und der Bewertungsfunktion. Der Zuggenerator erzeugt alle legalen Züge, die Suche probiert alle möglichen eigenen und gegnerischen Züge aus, und die Bewertungsfunktion entscheidet, welche Kombination für das Programm die beste Stellung ergibt. Das harmonische Zusammenspiel dieser drei Bestandteile bestimmt ganz wesentlich die Qualität eines Schachprogramms.

Das erste Problem, das der Schachprogrammierer lösen muß – und vielleicht das haarigste –, ist es, dem Computer die Stellung der Figuren auf dem Brett beizubringen. Denn natürlich kann der Zuggenerator nur dann einen Zug erzeugen, wenn diese Information, die sogenannte „Brettrepräsentation", in geeigneter Form vorliegt. Claude Shannon hatte bereits Ende der 40er Jahre eine einfache und heute klassische Brettrepräsentation vorgeschlagen. Sie ergänzt das 8×8-Schachbrett unten und oben mit jeweils zwei Reihen, sowie links und rechts mit jeweils einer Spalte. Es entsteht dadurch ein 10×12-Brett (Bild 1).

Da Computer eindimensionale Arrays in der Regel schneller als zweidimensionale adressieren, werden die Felder von 0 bis 119 durchnumeriert und in dieser Reihenfolge gespeichert



(Bild 1a). Um eine beliebige Stellung auf dem Schachbrett darzustellen, erhält jede Figur eine eigene Nummer: der weiße Bauer die 1, der weiße Springer die 2, ... , der weiße König die 6. Für die schwarzen Figuren werden dieselben Zahlenwerte verwendet, nur mit einem Minuszeichen davor: der schwarze Bauer erhält die -1 und so weiter. Ein leeres Feld hat den Wert 0, die Randfelder den Wert 100. Bild 1b zeigt die Ausgangsstellung in dieser Repräsentation. Steht auf dem Schachfeld b1 also ein Springer, so hat in der internen Darstellung das Feldelement 22 den Wert 2.

Die Zugmaschine

Der Computer bewegt nun eine Figur, indem er die Zahl des Ausgangsfeldes (Von-Feld) in das Zielfeld (Nach-Feld) überträgt. Zuvor muß jedoch sichergestellt sein, daß der Zug den Regeln des Spiels entspricht. Ein Beispiel: Ein Springer kann im allgemeinen von seiner Position aus acht Felder erreichen (Bild 2). Die Adressen dieser Felder gehen durch Addition der Zahlen 21, 19, 8, 12, -21, -19, -8 oder -12 zur momentanen Springer-Position hervor. In der Fachsprache heißen diese additiven Terme „Offsets". Steht der Springer also in Feld 64 (d5), so lauten die potentiellen Zugrichtungs-Adressen 85 (64+21), 83 (64+19), 72 (64+8), 76 (64+12), 43 (64-21), 45 (64-19), 56 (64-8) oder 52 (64-12).

Die so berechneten Züge heißen „pseudolegal", denn nicht immer sind sie zulässig. Das Programm muß noch weitere Voraussetzungen prüfen, wie etwa:

– Liegt die Zieladresse noch auf dem Brett?
– Läuft die Figur über die Brettkante hinaus?
– Ist das Zielfeld, auf das die Figur ziehen soll, leer?

Im Falle des Springers auf b1 liefern beispielsweise nur die Offsets 21 (b1-c3), 19 (b1-a3), und 12 (b1-d2) gültige Züge. Bei den übrigen landet der Springer auf einem mit 100 gekennzeichneten Randfeld. Ohne Randfelder würde der Springer beim Offset 8 den eher skurrilen Zug b1-h2 ausführen und bei den negativen Offsets das Programm wegen eines Zugriffs auf einen nicht deklarierten Datenbereich zum Absturz bringen. Durch die Einführung des Randes landet er immer auf einem definierten Feld. Bei den Zügen der weißen Figuren muß nicht mal der Rand abgefragt werden. Eine weiße Figur kann auf ein Nach-Feld ziehen, wenn dessen Wert kleiner gleich 0 ist (dann ist das Feld entweder leer oder es steht eine schwarze Figur auf dem Nach-Feld). Für die schwarzen Züge muß der Wert des Nach-Feldes größer gleich 0 und ungleich 100 sein.

Bauern- und Königszüge werden bis auf andere Offsets nach demselben Prinzip erzeugt. Bei den langschrittigen Figuren – Läufer, Turm und Dame – wird der Offset einer Richtung (beispielsweise 9 für die Läufer-Diagonale a1-h8) addiert, bis entweder ein Randfeld oder eine andere Figur erreicht ist. Danach werden – ausgehend von der Anfangsposition – die Züge für alle anderen, möglichen Richtungen erzeugt.

Um alle weißen/schwarzen Züge zu entwickeln, klappert der Zuggenerator das gesamte Feld ab. Findet er eine weiße/schwarze Figur, erzeugt er nach obiger Methode deren mögliche Züge und trägt sie in eine Liste ein. Zwei Spezial-Routinen kümmern sich anschließend noch um die Sonderfälle Rochade und Enpassant. Das war's dann aber auch schon. Ein nach diesem Schema gebauter Zuggenerator ist etwa 100 bis 200 Programmzeilen lang.

Das Pferd von hinten aufzäumen

Unser Schachprogramm kennt jetzt alle Regeln des Spiels. Es kann immer angeben, ob ein Zug zulässig ist oder nicht, und kann alle erlaubten Positionswechsel in Bruchteilen einer Sekunde ermitteln. Um nun noch den besten Zug auszuwählen, spielt es alle möglichen Fortsetzungen der Partie durch, bewertet die jeweilige Schlußstellung und rechnet dann bis zur aktuellen Position zurück. Dabei sucht es immer die optimale Fortsetzung, den besten Zug also, und geht von der grundlegenden Annahme aus, daß der Gegner dasselbe tut. Der Fachausdruck heißt „Minimax-Prinzip": Das Programm maximiert sein Ergebnis, und der Gegner versucht, es durch seinen besten Gegenzug zu minimieren.

Zur Berechnung des Minimax-Wertes für einen bestimmten Zug zäumt der Schach-Algorith-

110	111	112	113	114	115	116	117	118	119
100	101	102	103	104	105	106	107	108	109
90	91	92	93	94	95	96	97	98	99
80	81	82	83	84	85	86	87	88	89
70	71	72	73	74	75	76	77	78	79
60	61	62	63	64	65	66	67	68	69
50	51	52	53	54	55	56	57	58	58
40	41	42	43	44	45	46	47	48	49
30	31	32	33	34	35	36	37	38	39
20	21	22	23	24	25	26	27	28	29
10	11	12	13	14	15	16	17	18	19
0	1	2	3	4	5	6	7	8	9

100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100
100	-4	-2	-3	-5	-6	-3	-2	-4	100
100	-1	-1	-1	-1	-1	-1	-1	-1	100
100	0	0	0	0	0	0	0	0	100
100	0	0	0	0	0	0	0	0	100
100	0	0	0	0	0	0	0	0	100
100	0	0	0	0	0	0	0	0	100
100	1	1	1	1	1	1	1	1	100
100	4	2	3	5	6	3	2	4	100
100	100	100	100	100	100	100	100	100	100
100	100	100	100	100	100	100	100	100	100

Bild 1. Das „klassische" 10x12-Brett: Links (a) ist das Schachbrett mit den Speicheradressen dargestellt, rechts (b) mit den Inhalten der Adressen.

mus das Pferd quasi von hinten auf. Angenommen, Weiß ist am Zug: Dann erzeugt der Zuggenerator der Reihe nach alle möglichen weißen Züge und führt sie aus. Auf jeden weißen Zug folgt eine Unzahl schwarzer Gegenzüge, auf die dann wiederum eine Reihe von weißen Zügen folgt. Bei einer vorgegebenen Länge der Zugfolge, der sogenannten „Suchtiefe", wird dieser Prozeß abgebrochen. Eine Bewertungsfunktion belegt die resultierende End- oder „Horizontstellung" mit einer einzigen Zahl – eine diffizile Angelegenheit, zumal es dafür keine Formel, sondern nur Heuristiken oder Daumenregeln gibt.

Bild 3 zeigt eine Stellung und den zugehörigen Minimax-Suchbaum. Stellungen mit weißem Zugrecht sind als Kreise, solche mit schwarzem als Quadrate eingezeichnet. Die Zahlen unter den Horizontstellungen – 4, 5, 7 ... – sind das Ergebnis einer Bewertungsfunktion: Größere Zahlen bedeuten einen größeren Vorteil für Weiß und einen entsprechenden Nachteil für Schwarz. Aus Gründen der Übersichtlichkeit haben wir für jede Seite nur jeweils zwei Züge eingetragen.

Angenommen, vor dem Horizont ist Weiß (Kreis) am Zug. Weiß wird dann – dem Minimax-Prinzip folgend – in dieser Stellung denjenigen Zug spielen, der die höchste Endbewertung hat, also in Stellung 3 die 800, in 6 die 875, in 10 die 87 und in 13 die 1234. In Bild 3 sind diese Werte neben den Kreisen in Klammern eingetragen. Diese Ebene des Suchbaumes betrachten wir als neuen Horizont, und vor dem neuen Horizont ist diesmal Schwarz am Zug. Schwarz wird – dem Minimax-Prinzip folgend – in dieser Stellung denjenigen Zug spielen, der die niedrigste Endbewertung hat, also in Stellung 9 die 87 und in Stellung 2 die 800 (zur Erinnerung: je größer die Zahl, desto schlechter für Schwarz). In Bild 3 sind diese Werte neben den Quadraten in Klammern eingetragen. Und wiederum betrachten wir auch diese Ebene als neuen Horizont. Weiß wird diesmal von der Position 1 ausgehend die Stellung mit der Bewertung 800 bevorzugen. Damit steht der Zug, den der Schach-Algorithmus vorschreibt, fest.

Stellungs-Explosion

Das Minimax-Prinzip ist rekursiv und „watscheneinfach" zu programmieren. Einziges Problem: Ausgehend von einer Schachstellung sind im Mittel 36 Züge möglich, die ihrerseits

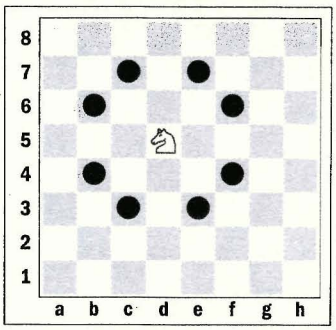
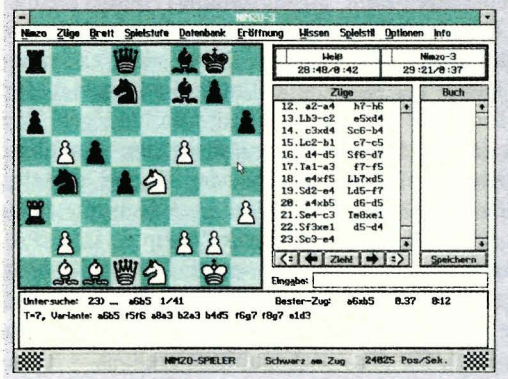


Bild 2. Legale Züge in Zelle 64 (d5) für den Springer: Die möglichen Offsets sind 21, 19, 8, 12, -21, -19, -8, -12.

mit jeweils 36 Zügen zu beantworten sind, auf die wiederum jeweils 36 Züge folgen. Selbst auf einem Pentium-Pro wäre spätestens nach fünf Zügen Sendepause, da das Programm mehr als 60 Millionen Endstellungen berechnen müßte.

Beschränkt durch eine „geringe Taktfrequenz", können Menschen pro Zug nur ein paar hundert mögliche Stellungen analysieren. Sie kompensieren dieses Manko, indem sie nur die „relevanten" Gegenzüge betrachten. Doch das Wörtchen „Relevanz" in Regeln zu fassen, ist gar nicht so leicht. Die Programmierer stellten fest, daß es praktisch unmöglich ist, einerseits schnell zu berechnende

NIMZO-3 – das Programm und die Erfolge



Die „Schaltzentrale" von NIMZO-3: Aus Effizienzgründen läuft das Programm im DOS-Fenster von Windows.

Einer der Autoren dieses Beitrags, Dr. Chrilly Donninger, ist der einzige hauptberufliche Schachprogrammierer im deutschsprachigen Raum. Sein liebstes Kind heißt NIMZO-3 – benannt nach dem in den 20er Jahren bekannten Schachspieler und Schachspiel-Theoretiker Aaron Nimzowitsch. Es ist nicht nur spielstark, sondern auch flexibel und kann neue Schachstrategien lernen oder fremde analysieren. Der Spieler kann das Verhalten von NIMZO-3 mittels der Schachsprache CHE! verändern. Und – last but not least – er kann auf eine Bibliothek von 100 000 Schacheröffnungen zurückgreifen.

NIMZO-3 ist nicht nur kommerziell ein Erfolg, sondern hat auch die folgenden Preise eingeheimst:

- 3. Platz Computerschach-Weltmeisterschaft 1996
- 2. Platz Computer-Blitzschach Weltmeisterschaft 1995
- Zusammen mit „Fritz" bestplaziertes Computerprogramm beim weltweit größten Mensch-Computer-Vergleichsturnier „Aegon" gegen Großmeister und Internationale Meister
- Sieger im „Carasaxa"-Blitzschachturnier 1996

und andererseits wasserdichte Regeln für die Auswahl der plausiblen Züge aufzustellen. Doch da Not bekanntlich erfinderisch macht, tüftelten gewiefte Schachprogrammierer den sogenannten „Alpha-Beta-Algorithmus" aus. Er ermittelt mit einem Bruchteil des Aufwands denselben besten Zug mit derselben Bewertung wie die oben beschriebene Minimax-Prozedur.

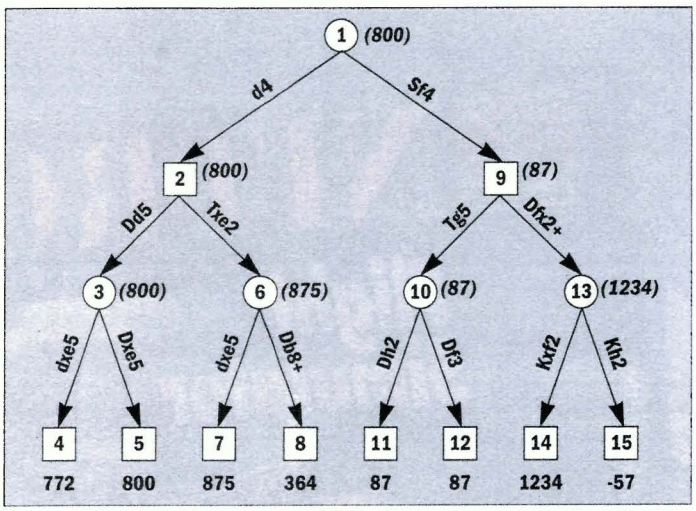


Bild 3. Ein Minimax-Spielbaum: Der Algorithmus wertet alle möglichen Züge aus und bewertet die Endstellungen.

Angenommen, Weiß (Kreis) ist am Zug und die Minimax-Bewertung des ersten Zuges ist bereits bekannt. In Bild 4 ist dies der Wert 800 in Stellung 3. Weiß kann sich mit einem der anderen Züge nur noch verbessern. Dieser erste garantierte Wert 800 ist die untere Schranke dieser Stellung, die „Alpha-Schranke". Umgekehrt ist dieser Wert für Schwarz eine obere Schranke, die „Beta-Schranke". Sei nun der zweite Zug und Stellung 6 betrachtet. Das Programm prüft den ersten Zug für Weiß und ermittelt für Stellung 7 den Wert 875. Die Überprüfung von Stellung 8 ist nicht mehr nötig: Schwarz, das in Stellung 2 am Zug ist, ermittelt einen Wert von 800 in Stellung 3. Für die Stellung 6 wird Schwarz ausrechnen, daß Weiß mindestens 875 erreichen kann. Schwarz wird daher die Stellung 3 herbeiführen.

Der Wert für Stellung 8 braucht deshalb gar nicht erst berechnet zu werden. Nach der gleichen Logik wird der rechte Ast abgearbeitet. Weiß ermittelt den Wert 87 für Stellung 10. Der Wert für Stellung 13 braucht nicht mehr berechnet zu werden, denn egal welchen Wert Stellung 13 hat: 87 ist der garantierte Wert für Schwarz in Stellung 9. Der Maximierer Weiß in Stellung 1 wird 800 präferieren. Eine optimale Alpha-Beta-Suche führt nur die schwarz gezeichneten Züge in Bild 4 aus, die farbig markierten Teile des Baumes werden auf Grund des Alpha-Beta-Kriteriums weggeschnitten und nicht untersucht. Listing 1 faßt den Alpha-Beta-Algorithmus in Pseudo-Pascal. Heute sinnieren allenfalls noch Universitätsprofessoren und Schach-Exweltmeister über intelligente, selektive Methoden nach. Programmierer nehmen diese Diskussion indes schon lange nicht mehr ernst.

Bestenfalls – der optimale Gegenzug wird als erster ausgeführt – untersucht der Alpha-Beta-Algorithmus nicht 36, sondern nur sechs Züge. Schlimmstenfalls – die schlechtesten Züge stehen am Anfang der Suche – degeneriert er zu Minimax. Die Qualität eines Schachprogramms hängt daher wesentlich davon ab, wie gut der Zuggenerator die Züge sortiert. Tatsächlich analysiert das von einem von uns (Chrilly Donninger) programmierte Schachprogramm NIMZO-3 (Kasten „NIMZO-3 – das Programm und die Erfolge") im Schnitt nur vier bis fünf Züge. Greift der Zuggenerator einmal daneben und klassifiziert einen gu-

Alpha-Beta-Algorithmus

```
1 AlphaBeta(p: position;alpha,beta,depth: integer) : integer;
2 var
3   score, w ,t ,m : integer;
4   moves: array[1..MAX_WIDTH] of integer;
5 begin
6   if depth = 0 then { Horizont erreicht }
7   return (Evaluate(p)); { Bewertungsfunktion aufrufen }
8   w := GenerateMoves(moves); { Zuggenerator aufrufen }
9   if w = 0 then { Keine Züge in dieser Position vorhanden }
10  return (Evaluate(p));
11  score := -999999;  { Sehr kleine Zahl. 'minus unendlich' }
12  for m:=1 to w do   { Alle Züge der Reihe nach ausprobieren }
13    begin
14      { Rekursiver Aufruf, die Rollen von Alpha,Beta werden
15        vertauscht, die Vorzeichen umgedreht, damit einheitlich
16        maximiert werden kann.}
17      t := -AlphaBeta(p.moves[m],-Beta,-Alpha,depth-1);
18      if t > score then { Verbesserung gefunden }
19        score := t;
20      if score >= beta then
21        { Beta-Cutoff. Widerlegung des letzten gegnerischen
22          Zuges gefunden. Weitere Züge brauchen nicht mehr
23          untersucht werden. }
24      return (score);
25      alpha := MAX(alpha,score);
26    end;
27  return (score); {Bewertung des besten Zuges}
28 end.
```

Deep Blue – Schlägt ein Silzium-Ungetüm den Schach-Weltmeister?

Der 1,4 Tonnen schwere Parallelrechner „Deep Blue" von IBM in der einen Ecke, der seit elf Jahren amtierende Schachweltmeister Gary Kasparow in der anderen: Der ganz in blau gekleidete, aber unsichtbare Schachcomputer tritt vom 3. bis 10. Mai zum zweiten Mal im Kampf „Chip gegen Gehirn" an. Austragungsort: New York City. Preisgeld: 700 000 Dollar. Wieder einmal soll das Turnier die Frage entscheiden, ob eine Maschine stark genug sei, den Weltmeister zu schlagen. Bislang konnten Schachcomputer zwar gegen Großmeister gewinnen, aber beim Weltmeister zogen sie immer den kürzeren.

Der Name Deep Blue ist an IBMs Spitznamen „Big Blue" angelehnt. Den meisten Experten gilt der Rechner als derzeit stärkste Schachmaschine. Das fünfköpfige Entwickler-Team hat eigens dafür einen speziellen Schach-Chip entwickelt. Während ein Teil des Prozessors aus der aktuellen Stellung alle möglichen Zugfolgen generiert, übernimmt der andere die Bewertung der entstehenden Positionen. Mehrere hundert dieser Chips sind auf dem Parallelrechner RS/6000 SP2 angeordnet. Deep Blue bewertet zirka 200 Millionen Positionen in der Sekunde oder 50 Billionen in den drei Minuten, die in einem Turnierspiel für einen Zug zur Verfügung stehen.

Bereits das erste Turnier im Februar letzten Jahres in Philadelphia, veranstaltet von der International Computer Chess Association, fand ein enormes Presseecho. Nachdem IBMs Wundermaschine den stärksten Schachspieler der Welt gleich im ersten Spiel in die Schranken gewiesen hatte, rettete Kasparow mit einem 4:2 Gesamtsieg doch noch die menschliche Ehre. In die Enge getrieben fühlte sich der Weltmeister des öfteren: „Nach so einem Spiel hat man erstmal eine schlaflose Nacht", meinte Kasparow nach seiner Anfangsniederlage. Für das diesjährige Turnier im Mai wurde die Maschine weiter verbessert: „Beim ersten Match haben wir die Technologie getestet, und wir haben eine Menge aus dem Duell gelernt", erklärt Deep-Blue-Teammanager CJ Tan. „Jetzt haben wir die Technik verbessert, und wir werden gewinnen."

Nähere Infos über die Veranstaltung unter:
http://www.chess.ibm.park.org.

ten Zug nicht als solchen, hat das aber im Gegensatz zur selektiven Methode keine katastrophalen Auswirkungen – die Suche dauert dann halt etwas länger.

Alpha-Beta verdoppelt die Suchtiefe gegenüber Minimax bei gleicher Rechengeschwindigkeit. Auf einem leistungsfähigen PC berechnen gegenwärtig gute Programme zehn bis elf Züge im voraus. Die sogenannten „scharfen Varianten", in denen viele Schachgebote oder Schlagzüge vorkommen, werden sogar noch tiefer berechnet. Der „Normalsterbliche" hat das Nachsehen. Maestros hingegen überblicken selbst verfahrene Situationen. Vor kurzem noch amüsierten sie sich über die Züge der Blechtrottel. Heute jedoch schauen sie schon finster drein, wenn der Programmierer auch nur ans Atmen denkt.

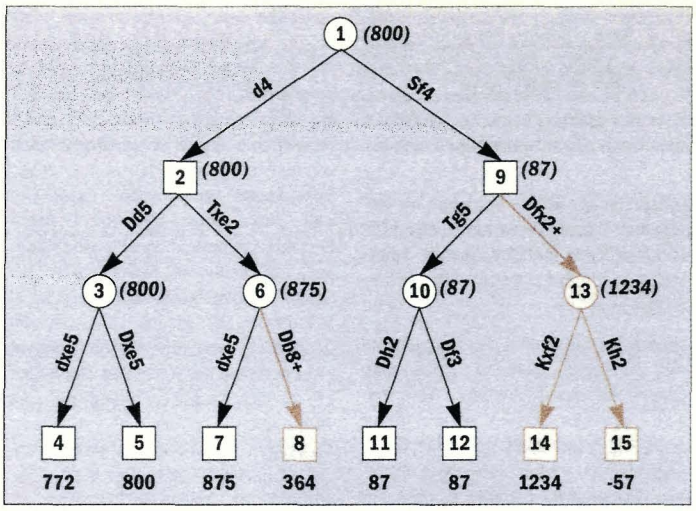


Bild 4. Spielbaum mit Alpha-Beta-Abschneidungen: Der Algorithmus bewertet nur die relevanten Spielzüge.

Fazit

Jeder nicht gänzlich untalentierte Tennisanfänger bringt mit ein wenig Übung den Ball übers Netz. Im Prinzip tut Boris Becker nichts anderes. Trotzdem war sein Aufstieg zum Wimbledon-Sieger lang und dornig. Für Schachprogrammierer gilt im Prinzip dasselbe. Es ist keine Hexerei, eine halbwegs brauchbare Software auf die Beine zu stellen. Um aber „Genius", „Fritz" oder „NIMZO" wegzuputzen, sind – wie bei jeder menschlichen Spitzenleistung – Talent und Programmierkenntnisse, vor allem aber Zähigkeit und Ausdauer unabdingbare Voraussetzungen. Im Gegensatz zum Tennis sind Computerschach-Profis aber noch nie mit Publicity belohnt worden. Weder die deutsche „Bild" noch die österreichische „Kronenzeitung" haben sich je für das Liebesleben eines Computerschach-Großmeisters interessiert.

Und es läuft das Gerücht, daß auch Fergie kein besonderes Interesse an Computerschach zeige. Immerhin saß bisher aber auch noch kein Computerschach-Großmeister wegen Steuerhinterziehung in Millionenhöhe hinter Gittern. Angesichts der Umsätze, die mit eher intellektuellen Spielen zu erreichen sind, wird sich dieses wohl auch in Zukunft nicht ändern. So betrachtet, ist Computerschach doch eher ein Spiel der Bettler als der Könige. *sk*

Literatur:

[1] *D. Steinwender, F. A. Friedel*: Schach am PC, Markt und Technik, Haar, 1995

[2] *M.M. Botwinnik*: Meine neuen Ideen zur Schachprogrammierung, Springer, Berlin, 1982

[3] *A. Reinefeld*: Spielbaum-Suchverfahren, Springer, Berlin, 1989

[4] *D. Levy, M. Newborn*: How Computers Play Chess, Freeman, 1991

[5] *P.H. Winston*: Künstliche Intelligenz, Addison-Wesley, Bonn, 1987

[6] *N.H. Yazgac*: Schachcomputer – Was sie wirklich können, Beyer-Verlag, 1989

Die besten PC-Schachprogramme

Die folgenden Schachprogramme agieren auf Großmeister-Niveau. Einige, NIMZO-3 beispielsweise, verzichten aus Effizienzgründen auf Windows. Ihre Spielstärke wird in Elo ausgedrückt, zum Vergleich: Schachanfänger 1000-1200, Landesliga-Spieler 2000-2200, Großmeister ab 2500, Weltmeister 2800 Elo. Alle Programme sind in [1] ausführlich beschrieben.

Dort finden Sie auch weniger leistungsfähige Schach-Shareware. Kommerzielle Software wird von Fachhändlern angeboten (zum Beispiel Schach & Spiel, Berlin, Tel. 030/3425820). Im Internet gibt es aktuelle Informationen und Bestellmöglichkeiten unter

`http://www.dss.de/schach.html`

und

`http://www.gambitsoft.com`

Die letzte Web-Seite ist gleichzeitig eine interessante Schach-Page mit weiterführenden Links.

Programm	Chess Genius 5	Chessmaster 5000	Fritz 4	Hiarcs 4	MChess Pro 5	NIMZO-3	Rebel 8
Funktion	Schachprogramm	Schachprogramm	Schachprogramm	Schachprogramm	Schachprogramm	Schachprogramm	Schachprogramm
Hersteller	Richard Lang, Dorset, England	Software Toolworks, Burgess Hill, England	Chess Base, Hamburg	Applied Computer Concepts, England	Marty Hirsch, USA	Dr. Christian Donninger, Wien	Schröder BV, Niederlande
Autor	Richard Lang	Johan de Koning	Franz Morsch	Mark Uniacke	Marty Hirsch	Dr. Christian Donninger	Ed Schröder
Systemvoraussetzungen	386er, 8 MByte RAM	486er, 4 MByte RAM	386er, 640 KByte	386er/2 MByte	386/640 KByte	386/4 MByte	386/4 MByte
Spielstärke	2400 Elo (Pentium, 90 MHz)	2310 Elo	2300 Elo	2320 Elo	2652 Elo (Aegon-Turnier)	2598 Elo (Aegon-Turnier)	2480 Elo (Pentium, 90 MHz)
Preis	198 Mark	99 Mark	198 Mark	198 Mark	198 Mark	198 Mark	198 Mark
Kommentar	Derzeit eines der stärksten Programme auf dem Markt. Bietet strategisch gutes Schach.	Nur Englisch, aber preisgünstig.	Marktführuer und herausragender Taktiker. Kann Spieler bewerten.	Wird auch als Mac-Version angeboten.	Etwas schwächer als Genius. Aktiver und dynamischer Stil.	Spielstarkes DOS-Programm, läuft auch unter Windows.	Hohe Spielstärke mit druckvollem, dynamischen Spiel.

„Ein ruhiger Hintern ist wichtiger als ein genialer Geist“

Dr. Christian „Chrilly“ Donninger ist Wiener, promovierter Mathematiker und Trachten-Fan. Er legt Wert auf die Tatsache, daß er aus Oberösterreich stammt. Schon seit seiner Kindheit löst er mit Leidenschaft logische Rätsel. Sein geistiges Kind NIMZO ist der Hit in der Computerschach-Szene. Donninger versteht sich als traditioneller Handwerksmeister.

Dr. Klaus Manhart ist studierter Wissenschaftsphilosoph und in München als Redakteur tätig. Beide kennen sich aus gemeinsamen Wiener Tagen am „Institut für Höhere Studien“, der österreichischen „Hochburg“ der mathematischen Spieltheorie. Manhart führte das hier abgedruckte Interview mit Donninger im Auftrag von mc extra.

mc extra: Herr Donninger, Sie haben eines der spielstärksten Schachprogramme – NIMZO-3 – entwickelt. Was ist Ihr Schlüssel zum Erfolg?

Donninger: Das Erfolgsgeheimnis ist dasselbe wie in anderen Sportarten und eigentlich allen Gebieten menschlicher Tätigkeit. Man muß ein gewisses Talent und technische Fähigkeiten mitbringen. Das wichtigste ist aber Ausdauer und Fleiß, oder – wie ich es gerne nenne – ein ruhiger Hintern.

mc extra: Wie lange arbeiten Sie schon an Ihrem Programm?

Donninger: In NIMZO-3 stecken etwa anderthalb Entwicklungs-Jahre oder rund 3000 Stunden. Die schließen nicht nur den Algorithmus, sondern auch die Oberfläche, die Datenbank und die Schach-Programmiersprache CHE! mit ein. Der Code von NIMZO-1 und NIMZO-2 ist nicht in den von NIMZO-3 geflossen, aber natürlich die dabei gewonnenen Erkenntnisse. Dafür könnte man nochmal 2000 Arbeitsstunden veranschlagen.

mc extra: In welche Richtung entwickeln Sie NIMZO weiter?

Donninger: Eigentlich sind die Programme für die meisten schon viel zu stark. Wichtiger sind momentan die Verpackung und die Gags. Ich stelle gerade ein Multimedia-Programm mit dem Namen „Champ“ fertig. Es enthält Videosequenzen, in denen ich meinen Senf zum Spielgeschehen abgebe – in Lederhose, so wie sich halt der Piefke den Ostmärkler vorstellt. Danach möchte ich aber wieder was Seriöses tun. In NIMZO-4 werde ich vermehrt Lernregeln einbauen. Das Programm soll sowohl aus den Partie-Sammlungen von Meisterspielern, als auch aus seinen eigenen Partien mehr und besser als bisher lernen.

mc extra: Gab es schon mal einen direkten Vergleich mit Kasparow?

Donninger: Um gegen Kasparow spielen zu können, muß man nicht nur ein sehr gutes Programm, sondern auch viele, viele Dollars besitzen. Da ich nur ersteres habe, kam es bisher noch zu keiner derartigen Auseinandersetzung. Außerdem glaube ich, daß Kasparow gegen Computer keineswegs der beste Spieler ist. Sein aggressiver Stil kommt den Programmen sehr entgegen. Computer muß man durch gekonntes Nichtstun zu Fall bringen. Zudem ist Schach, so wie Boxen auch, ein psychologischer Krieg Mann gegen Mann – das ist das größte Handikap für weibliche Spieler. Vor allem in dieser Hinsicht ist Kasparow Weltmeister.

mc extra: Was zeichnet Ihrer Meinung nach einen guten Schachprogrammierer aus?

Donninger: Erstens muß man sehr gut programmieren können. Dabei verstehe ich unter „gut Programmieren“ auch die Fähigkeit, die Probleme so umzuformulieren, daß sie der Blechtrottel möglichst effektiv bearbeiten kann. Außerdem darf man kein Perfektionist sein. Praktisch alle Computerschach-Techniken sind Heuristiken, die sich meistens vorteilhaft, manchmal aber auch katastrophal auswirken. Nebensächlich, ja sogar schädlich, sind sehr gute Schachkenntnisse. Überragende Schachspieler denken in allzu komplexen Begriffen. Wichtig ist aber: Wie sage ich es dem Blechtrottel möglichst einfach?

mc extra: Was halten Sie von Deep Blue, und worin sehen Sie den Unterschied zu Ihrem Programm NIMZO-3?

Donninger: Das Deep-Blue-Team hat von Anfang an so ziemlich alle Schach-Feinheiten zugunsten der Effizienz geopfert. Deep Blue besteht aus ein paar hundert speziellen Chips, die „nur“ Schachspielen können. Dadurch ist die Maschine etwa tausendmal schneller als NIMZO-3 auf einem 200 MHz Pentium-Pro. Das Programm ist auf Grund seiner Geschwindigkeit taktisch natürlich sehr stark, es kann dafür aber so gut wie überhaupt nicht Schach spielen. Im Verhältnis zum ungeheuren Aufwand und den vollmundigen Versprechungen – der Weltmeister sollte schon 1992 endgültig besiegt werden – sind die Erfolge eher bescheiden. Das Deep-Blue-Team spielt nicht gern gegen andere Programme. Gewinnt Deep Blue, heißt es „na ja, kein Wunder bei der Hardware-Überlegenheit“. Verliert es, sind sie das Gespött der Computerschach-Szene. Meiner Meinung nach hätte NIMZO-3 auf einem Pentium Pro ganz gute Gewinnchancen gegen Deep Blue.

„Mein einziges Kriterium ist, daß die von mir hergestellten Produkte keinen Schaden anrichten und nicht übermäßig zur weiteren Verblödung der Menschheit beitragen.“

mc extra: Psychologen und Schachexperten werfen heutigen Schachprogrammen vor, daß sie – statt den menschlichen Denkprozess nachzubilden – rohe Gewalt, „brute force“, einsetzen. Glauben Sie, daß irgendwann wieder eine Rückkehr zur „kognitiven Modellierung“ bei Schachcomputern einsetzt?

Donninger: Ich glaube, das Beispiel Deep Blue zeigt, daß Geschwindigkeit nicht der Stein der Weisen ist. Deep Blue hat nämlich dasselbe Problem wie „der Wülde auf seiner Maschin“ von Helmut Qualtinger: „I was zwar net wo I hin wü, aber dafür bin I schneller durt“. Eine Rückkehr zu „kognitiven Methoden“ in dem Sinn, daß man versucht den menschlichen Denkprozeß nachzubilden, wird es aber mit Sicherheit auch nicht geben. Erstens weiß man sehr wenig, was sich in einem Hirn abspielt und zweitens ist das menschliche Denken wohl für den Aufbau und auch die Beschränkungen der biologischen „Hardware“ optimiert.

„Deep Blue hat dasselbe Problem wie ‚der Wülde auf seiner Maschin‘ von Helmut Qualtinger: ‚I was zwar net wo I hin wü, dafür bin I schneller durt‘“

mc extra: Wie schätzen Sie die zukünftige Entwicklung bei der Schachprogrammierung ein?

Donninger: Meiner Meinung gibt es zwei vielversprechende Richtungen der Weiterentwicklung. Erstens lassen sich die bisherigen „Brute-Force“-Methoden noch verbessern. Die heutigen Programme sind taktisch zwar sehr stark, allerdings sind Großmeister noch durchaus in der Lage, die Programme in Stellungen mit bekannten Mustern auch taktisch auszurechnen. Eine Richtung der Weiterentwicklung besteht darin, daß die Programme, so wie die Großmeister, besonders interessante Varianten wesentlich weiter berechnen als die übrigen. Das tun sie schon heute, allerdings noch nicht gut genug.

Es sind heute fast alle wichtigen Schachpartien in maschinenlesbarer Form vorhanden. Man müßte daher „nur“ einen Weg finden, wie das Programm auf Grund dieser Partien seinen eigenen Spielstil verbessert. Natürlich sollte es auch aus seinen eigenen Erfahrungen – sprich Niederlagen – lernen. Diese Lernmethoden werden am „Brute-Force“-Stil nichts verändern, es geht vielmehr darum, der Suche halbwegs intelligente Ziele vorzugeben.

mc extra: Sie verbringen zehn bis zwölf Stunden täglich mit Schachprogammierung. Was fasziniert Sie persönlich so sehr daran?

Donninger: Ich spiele selbst Schach zur Unterhaltung, aber es fasziniert mich nicht mehr als andere Spiele wie beispielsweise Tarock oder Backgammon. Ich bin Mathematiker und löse eigentlich schon seit frühester Kindheit mit Leidenschaft logische Rätsel. Zum Beispiel hat mir meine Mutter im Alter von vier Jahren einen Bleistift zum Zeichen gekauft. Anstatt damit die Wände der elterlichen Wohnung zu bekritzeln, habe ich mich hingesetzt und das auf dem Bleistift aufgedruckte kleine Einmaleins auswendig gelernt. Mit Computerschach habe ich ein dankbares Problem gefunden, von dem ich auch halbwegs leben kann. Ich würde aber mindestens genauso gerne einen möglichst guten und schnellen Java-Interpreter für Windows-95 schreiben.

Es gibt noch einen zweiten Punkt, der mich fasziniert: Schach ist zwar schwierig, aber dennoch überschaubar. Es ist zu diffizil für den menschlichen Geist – aber nur ein bißchen.

mc extra: Sehen Sie Ihre Tätigkeit als Produzent von Unterhaltungsspielen oder verbinden Sie eine weitreichendere Philosophie damit?

Donninger: Ich fühle mich als traditioneller Handwerksmeister, der Stolz auf seine handwerklichen Fähigkeiten und auch auf das von ihm zur Gänze selbst hergestellte Produkt ist. Nachdem ich von meinen Produkten auch lebe, muß ich mich natürlich auch nach dem Geschack der Kundschaft und nicht nur nach meinem eigenen richten. Dasselbe tut aber auch ein Maßschneider oder ein Kunstschmied. Die Vorbilder bei meinem Schritt ins Schach-Computer-Profitum waren ein befreundeter Weinbauer und mein ebenfalls als Weinbauer tätiger Schwiegervater. Ihr Stolz auf das von ihnen hergestellte Produkt hat mich beeindruckt. Dieses Gefühl ging mir bei meiner Arbeit als Software-Entwickler bei der Firma Siemens und bei der Europäischen Raumfahrt gänzlich ab.

Für meine Arbeit habe ich mir eine eigene kleine Wohnung angemietet, die ich morgens um 7 Uhr betrete, von 12 bis 14 Uhr halte ich Siesta, danach geht es bis 20 Uhr weiter. In der „Werkstatt“ gibt es nicht einmal eine Klingel, geschweige denn Telefon, Fax oder Internet-Anschluß. Ich kann daher, so wie im Mittelalter, in meinem eigenen Tempo und trotz des großen Arbeitsumfanges weitgehend ohne Streß vor mich hin arbeiten. Das Philosophische meiner Arbeit besteht also in einem gewissen „Lob der Langsamkeit“, wobei diese Langsamkeit wesentlich produktiver ist als die oft inhaltsleere Hektik der postmodernen Welt. Mit meinen Produkten verbinde ich keine weihevollen Absichten. Mein einziges Kriterium ist, daß die von mir hergestellten Produkte keinen Schaden anrichten und nicht übermäßig zur weiteren Verblödung der Menschheit beitragen.