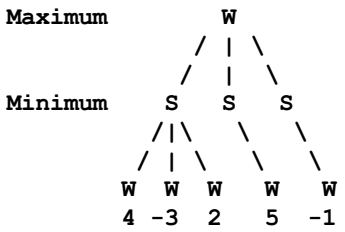
## Ahnungslos gute Züge finden

**Eigentlich sind die Erfolge der heutigen Schachprogramme unbestreitbar: 2006 schlug Fritz Weltmeister Kramnik in einem Match über sechs Partien mit 4:2 und erst vor kurzem verlor Großmeister Ehlvest in einem Schnellschach-Match gegen Rybka 2,5 : 5,5, obwohl der Mensch in diesem Falle sogar von Anfang an einen Bauer mehr auf dem Brett hatte.Jeder, der allerdings versucht hat, aus der Eröffnung heraus im Analysemodus eine Variante zu verfolgen, indem er eine starke Schach-Engine gegen sich selbst hat spielen lassen, wird bemerkt haben, dass die Vorschläge dieser Engine nicht zu dem Ergebnis führten, das laut Zugbewertung vorheriger Züge eigentlich zu erwarten gewesen wäre. Denn sehr oft stellen sich Züge im nachhinein als schlecht heraus, die anfangs von der Engine als gut eingeschätzt wurden. Woher kommt das?**

Das Programm ist dem sogenannten "Horizont-Effekt" zum Opfer gefallen, der schon seit Urzeiten allen Engines zu schaffen macht. Um diesen zu erklären, lohnt es sich, genauer auf die Methode zu schauen, wie ein Schachprogramm Züge findet.

Schon im Jahre 1950 veröffentliche Claude Shannon das Minimax-Verfahren, das auf der Tatsache beruht, dass es sich beim Schach um ein Spiel mit vollständiger Information handelt, d.h. der Zufall spielt keine Rolle, und beide Spielteilnehmer kennen während der ganzen Partie alle zum Spiel notwendigen Informationen. Der spieltheoretische Satz, daß in solchen Zweipersonenspielen die Summe der Gewinne (d.h. Vorteile) und Verluste (d.h. Nachteile) immer gleich Null ist, wurde schon 1928 von John v. Neumann publiziert. Ein guter Zug bedeutet also, dass es für den Gegner in allen möglichen Varianten nur schlechtere Züge gibt, die allesamt zu vorteilhaften und für den Gegner nachteiligen Stellungen führen. Jetzt muß "nur" noch geklärt werden, was im Schach eine vorteilhafte Stellung ist, und schon kann man auf einem internen Brett einfach die Züge in folgender Art und Weise ausprobieren:

Angenommen, Weiß ist am Zuge. Dann führt man nacheinander seine Züge aus, und für jeden schwarzen Antwortzug führt man wiederum alle weißen Anwortzüge aus, und so weiter, bis zu einer bestimmten Tiefe, in der z.B. Weiß am Zug ist.




Wie man an diesem Diagramm erkennt, hat Weiß in der Ausgangsstellung drei Züge. Dies führt im ersten Fall zu drei möglichen Antwortzügen von Schwarz, in den anderen Fällen hat Schwarz nur einen Zug. In typischen Schachpositionen gibt es natürlich viel mehr, nämlich durchschnittlich ca. 35 Antwortzüge, aber hier geht es nur erst einmal ums Prinzip. Dieses Diagramm stellt also einen Spielbaum dar, der mit der Ausgangsposition beginnt und mit fünf möglichen Blattknoten, d.h. Endpositionen, endet.

Für jeden der möglichen Blattknoten berechnet die sogenannte Bewertungsfunktion des Schachprogramms eine Zahl, die die Güte der jeweiligen Stellung anzeigt, also (zum Beispiel) positive Werte für gute weiße Stellungen und negative Werte für gute schwarze Stellungen.

Nach dem Minimaxprinzip muss nun Weiß, wenn er den "besten" Zug machen will, davon ausgehen, dass auch Schwarz den für sich "besten" Zug aussucht, d.h. Schwarz minimiert für sich seine Zugwerte und Weiß maximiert seine Zugwerte, also sucht Schwarz sich jeweils das Minimum aus. In dem obigen Diagramm-Beispiel würde Schwarz sich im ersten Unterbaum den Zug aussuchen, der zur Stellung mit Wert -3 führt, im mittleren und im rechten Unterbaum die Züge mit den Werten 5 und -1. Weiß jedoch bildet das Maximum dieser Werte, also von -3, 5 und -1 wäre das Maximum 5 und die Entscheidung fiele auf den mittleren Unterbaum. In einem Schachprogramm bekäme also die Hauptvariante den Wert +5.

Im Schach wird die Anzahl der Blattknoten des Spielbaums schnell sehr groß, sie "explodiert" exponentiell. Von der Größenordnung her wären das bei n Halbzügen schon $n^{35}$ Blattknoten. Also war es von entscheidender Bedeutung für die Praxis, dass es durch Sortierung der Züge nach (wahrscheinlicher) Qualität (z.B. erst Schlagzüge, dann Angriffszüge usw.) eine Möglichkeit gibt, ganze Unterbäume abzuschneiden und so weniger Endknoten bewerten zu müssen. Das Verfahren nennt sich Alpha-Beta-Suche und wird unter anderem in einem Wikipedia-Artikel näher erklärt. Für diese Betrachtung ist erst einmal wichtig, dass dadurch das Minimax-Verfahren im Kern nicht verändert wird.

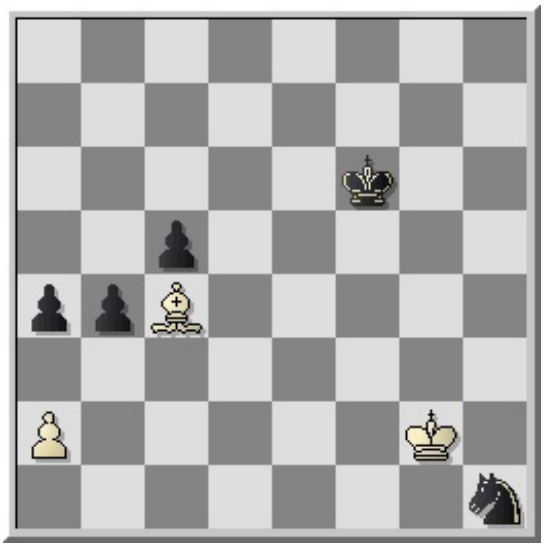
Die Qualität heutiger Schachprogramme wird also hauptsächlich durch die Art und Weise der Bewertungsfunktionen bestimmt, denn hier entscheidet es sich, was das Programm als gute Stellung oder als schlechte Stellung ansieht. Anfangs gab es da nur wenige Kriterien, vor allem das Abzählen des Materials und Bestimmung von Matt und Patt war üblich, aber man stellte sehr schnell fest, dass das bei weitem nicht ausreichend ist. So gehen heute solche Faktoren wie die Königssicherheit, Bauernstrukturen und Raumvorteil in die Bewertung ein, um nur einige zu nennen.

## Der Horizonteffekt

Nun wird das Phänomen des oben erwähnten "Horizonteffekt" klarer. Denn für ein Schachprogramm bilden die Blattknoten des Spielbaums den Horizont, hinter dem es nur in Ausnahmefällen noch weiß, was danach auf dem Brett passieren wird. Eine Ausnahme bildet hier die sogenannte "Ruhesuche", die immer dann, wenn ein Blattknoten durch Schlagen einer Figur oder eines Bauern entsteht, diese Variante verlängert, indem hier alle Schlagzüge soweit verfolgt werden, bis ruhige Stellungen ohne Schlagabtausch erreicht sind. Dahinter steckt die klare Absicht, Materialverlust zu vermeiden.

Als Illustration des Horizonteffekts betrachte man z.B. folgende Stellung, die der Dokumentation von Salomon Chess entnommen wurde:

FEN: 8/8/5k2/2p5/ppB5/8/P5K1/7n b - - 0 1



Wie man nach etwas genauerem Hinsehen bemerkt, ist der beste Zug von Schwarz 1...Ke5 mit der Absicht, den König an die eigenen Bauern am Damenflügel heranzuführen, um den Durchmarsch eines Bauern zur Dame zu erzwingen. Wenn man aber jetzt die Spielstufe der Engine auf feste Suchtiefe mit der Länge 4 Halbzüge einstellt, macht jedes nach der Minimax-Methode arbeitende Programm (und das sind fast alle) den schlechten Zug 1...b3. Was ist passiert? Es werden unter anderem die Varianten gefunden, in denen Weiß entweder den Bauern oder den Springer schlägt. Nach der Minimax-Methode ist aber das Schlagen des Springer in diesem Moment für Weiß besser, denn weiter wird hier nicht gerechnet, und nach 4 Halbzügen herrscht für das Programm tiefste Dunkelheit; deswegen der Missgriff, der es dem Weißen ermöglicht, doch noch ein Remis zu erreichen.

Im Prinzip agiert also jedes Schachprogramm wie ein Blinder, der sich mit seinem Krückstock an der Bordsteinkante entlangtastet und erst im Nachhinein merkt, dass etwas Entscheidendes passiert ist, z.B. dass jetzt der Bordstein zu Ende und er an einer Kreuzung angekommen ist. Weil es mit diesem Verfahren wegen der ungeheuren Anzahl der Züge ohne Zusatzinformationen fast unmöglich ist, aus der Eröffnung heraus in eine gute Mittelspielstellung zu gelangen, benutzen alle Schachprogramme sogenannte Eröffnungsbibliotheken, in denen gemäß der bekannten Eröffnungstheorie für beide Seiten spielbare Varianten vermerkt sind.

Aus dem gleichen Grund werden Endspieltabellen benutzt, in denen durch vorhergehende monatelange Berechnungen für alle Positionen mit (sehr) wenigen Steinen auf dem Brett alle Züge samt Gewinnmöglichkeiten gespeichert wurden, so daß direkt abgelesen werden kann, ob ein bestimmter Zug gewinnt, und wenn ja, in wieviel Zügen.

In allen Fällen, in denen ein Schachprogramm nicht in einer Tabelle nachschauen kann, werden zwangsläufig unglaublich viele unsinnige Zugkombinationen betrachtet. Was auf den ersten Blick wie pure Zeitverschwendung aussieht, hat aber auch zur Folge, dass innerhalb des betrachteten Spielbaums, d.h. diesseits des Horizonts, so gut wie keine taktischen Manöver übersehen werden. Diese Tatsache trägt entscheidend zum beträchtlichen Erfolg der Schachprogramme über menschliche Gegner bei.

Damit nicht Positionen, die bei der doch allerdings recht planlosen Suche schon einmal aufgetreten sind, doppelt bewertet werden, gibt es intern die sogenannten Hashtabellen, in denen Zwischenergebnisse abgespeichert werden und die sich der Anwender beim Analysieren zu Nutze machen kann. Aber dazu später mehr.

## Die Bewertungsfunktion und deren Grenzen

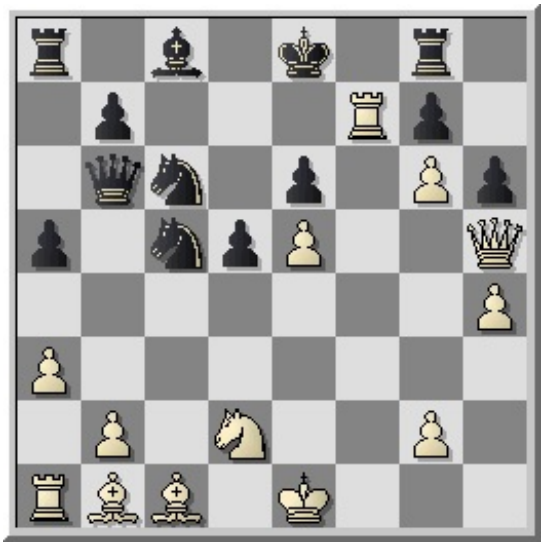
Die Qualität eines Schachprogramms, das nach der Minimax- bzw. Alpha-Beta-Methode arbeitet, läßt sich also auf die Qualität seiner Bewertungsfunktion reduzieren. Hier stehen die Programmierer aber vor einem nicht unerheblichen Problem: Sie müssen an Hand der auf dem Brett sichtbaren Stellung abschätzen, welche Seite besser steht, und das für alle Phasen des Spiels, auch wenn Materialgleichheit herrscht. Oft ist es sogar schwerer, einen Vorteil einzuschätzen, wenn das Material nicht ausgeglichen ist. Dabei gilt es, alles auf einen einzigen Zahlenwert abzubilden.

Die reinen Materialwerte der Steine auf dem Brett sind ja noch leicht aufzurechnen, aber es müssen dann Pluspunkte für positionelle Vorteile wie offene Turmlinie oder Springervorposten bzw. Strafpunkte für rückständige Bauern oder einem ungeschützen König verteilt werden. Gerade beim Thema der Königssicherheit wird das auftauchende Dilemma klar: Während vor allem im Mittelspiel der König möglichst durch Bauern gesichert sein sollte, wird im Endspiel der König zu einer aktiven Figur, für die es schlecht ist, wenn sie am Rande verharrt. Wann aber beginnt jetzt genau das Endspiel und kann man das nur durch das Material auf dem Brett definieren? Es gibt noch mehr solcher Gewissensfragen, denn um am Ende auf eine Zahl für den Stellungswert zu bekommen, müssen auch dynamische Elemente wie Figurenbeweglichkeit mit statischen Elementen wie dem Materialwert in Einklang gebracht werden.

Dabei hat sich im Laufe der Zeit herausgestellt, dass Programme mit ausgefeilten positionellen Bewertungsfunktionen besser abschneiden, besonders deutlich wurde das mit dem Erscheinen Rybkas. Trotzdem ist fraglich, ob diese der Weisheit letzter Schluss sind, man betrachte z.B. die Stellung, die in der Tarrasch-Variante der Französchen Verteidigung nach folgenden Zügen (entnommen einer Analyse des Autors) auf dem Brett auftaucht:

**1.e4 e6 2. d4 d5 3. Sd2 Sf6 4. e5 Sfd7 5. f4 c5 6. c3 cxd4 7. cxd4 Sc6 8. Sgf3 f6 9. Ld3 fxe5 10. dxe5 Sc5 11. Lb1 Le7 12. a3 a5 13. h4 Db6 14. Sg5 Kf8 15. Dh5 Lxg5 16. fxg5 Ke7 17. g6 h6 18. Tf1 Tg8 19. Tf7+ Ke8**

FEN: r1b1k1r1/1p3Rp1/1qn1p1Pp/p1npP2Q/7P/P7/1P1N2P1/RBB1K3 w Q - 0 20



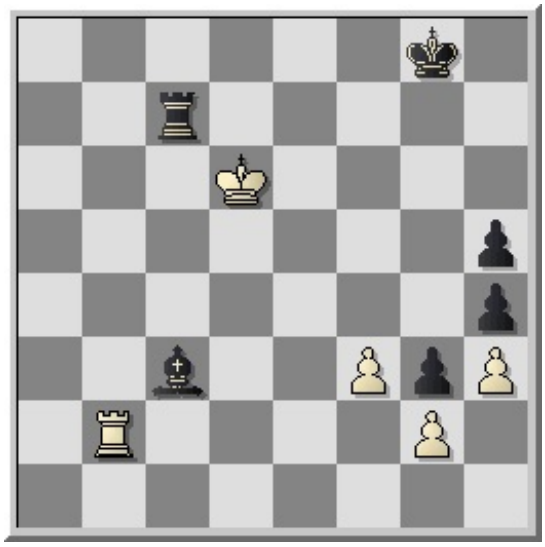
Besonders die modernen Engines meinen jetzt, dass Schwarz etwas besser steht und wollen überwiegend 20. De2? spielen. Dabei hat Weiß hier eine eindeutige Gewinnvariante nach **20. Txg7!**, z.B.

**20. Txg7 Txg7 21. Dxh6 Tg8 22. Dh7 Se7 23. Sf3 Kd7 24. Lg5 Te8 25. g7 Dd8 26. Lg6 Tg8 27. Tc1 b6 28. b4 axb4 29. axb4 Sa6 30. Ld3 Sc7 31. Lxe7 Kxe7 32. Txc7+ Dxc7 33. Dxg8 Dc1+ 34. Kf2 Df4 35. Dh7 Ta2+ 36. Le2 Df7 37. g8=D 1-0**

Zugegeben, der Vorteil von Weiß wird hier erst nach fünf oder sechs Zügen deutlich, aber moderne Engines erreichen diese Suchtiefe sehr schnell. Was also passiert hier? Dadurch, dass positionelle Faktoren die Bewertungsfunktion beherrschen, werden Opfer hinten in die Liste der zu untersuchenden Züge eingereiht. Bei der Suche wird inkrementell vorgegangen, d.h. die Suchtiefe wird langsam, Halbzug um Halbzug erhöht, um jeweils aus der vorhergehenden Untersuchung Anhaltspunkte für eine neue Sortierung in der laufenden Suchtiefe zu gewinnen. Da der Wert des Opfers relativ lange Zeit nicht sichtbar wird, ist es durchaus möglich, dass Txg7 durch eine Abschneide-Operation bei der Alpha-Beta-Suche übergangen wird. In diesem Sinne hätte also die Orientierung der Bewertungsfunktion hin zu mehr positionellen Faktoren das Durchrechnen eines auf den ersten Blick unsinnigen Zuges unmöglich gemacht. Interessanterweise finden ältere Programme wie Fritz 9 und sogar Fritz 10 noch das Turmopfer, während bei Fritz 11 dasselbe Phänomen wie bei Rybka auftritt, nämlich das Übersehen des Zuges.

Eine weitere interessante Stellung, die vor allem die Schwäche der Programme im Endspiel aufzeigt, ist die folgende:

FEN: 6k1/2r5/3K4/7p/7p/2b2PpP/1R4P1/8 b - - 0 1

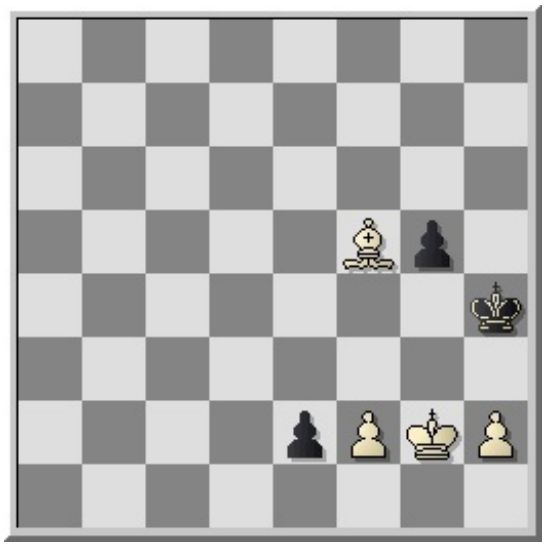


Dem schachspielenden Menschen fällt relativ schnell auf, dass das Nehmen des Turmes auf b2 einen Turmtausch zur Folge hat, der Schwarz zwar einen Mehrläufer beschert, aber danach ziemlich ratlos zurückläßt; denn wie soll der schwarzfeldrige Läufer einen der Bauern schlagen, die auf weißen Feldern stehen? Die einzige Chance hätte der schwarze König, aber der wird vom weißen König daran gehindert, nach g2 zu kommen, der Achillesferse des weißen Spiels. Der richtige Gewinnplan besteht also darin, auf den Turmtausch zu verzichten und den Läufer nach f2 zu bringen, um den Bauern g2 von der Deckung des Turms abzuschneiden.

Die meisten Engines nehmen aber bedenkenlos den Turm auf b2 und landen dann nach 2. Kxc7 im Remis. Der kurzfristig zu erreichende Materialvorteil überwiegt den längerfristigen Gewinnweg, der nur mittels bestimmter Berechnungen in der Bewertungsfunktion überhaupt entdeckt werden kann. Denn das Alpha-Beta-Verfahren muß den Erfolg bzw. Misserfolg erst einmal bemerkt haben, und das kann bei der Anzahl und Länge der auszuschliessenden Varianten lange dauern. Übrigens scheint Zappa Mexico II hier mehr zu tun als andere, denn dieses Schachprogramm fand dem Schlüsselzug **1... Tc4** ziemlich schnell, brauchte aber länger, um das siegbringende Manöver **1... Tc4 2. Te2 Kf7 3. Ta2 Td4+ 4. Kc5 Td1 5. Kc4 Le1 6. f4 Lf2** zu finden.

Was aber total jenseits der heutigen Fähigkeiten der Schachprogramme liegt, ist das Entdecken einer Festung. Eine Stellung mit einer Festung zeichnet sich dadurch aus, dass eine Figurenkonfiguration existiert, die der Gegner trotz überlegenen Materials nicht "knacken" kann, d.h. der Gegner müsste schon seinen Materialvorsprung aufgeben, um irgendeinen Fortschritt im Spiel zu erreichen; aber danach wäre nur noch ein Remis oder sogar der Gewinn möglich. Das bedeutet, dass Stellungen mit einer Festung remis sind. Hier ein Beispiel:

FEN: 8/8/8/5Bp1/7k/8/4pPKP/8 w - - 0 1



Nach **1. Lg4 e1D 2. h3** ist der schwarze König eingeklemmt und die schwarze Dame kann alleine nichts ausrichten, da es völlig ausreicht, wenn der Läufer zwischen f3 und g4 hin- und herzieht. Ein Mensch würde direkt Remis anbieten, aber ein Schachprogramm würde erst nach 50 Zügen (durch den Eingriff der 50-Züge-Regel) das Remis bemerken.

## Fazit

Hier wird die größte Schwäche aller heutigen Schachprogramme deutlich: Es ist das planlose Ausprobieren von Zügen mit der Hoffnung, dass schon etwas entdeckt wird, das dann ausgenutzt werden kann. Es werden größtenteils unsinnige Zugfolgen untersucht, was viel Zeit kostet und nur auf Grund der Schnelligkeit heutiger Prozessoren überhaupt durchführbar ist. Trotz dieser Rechenleistung können sehr lange Varianten absolut nicht bewältigt werden, aber gerade im Endspiel kommen Mattführungen vor, die extrem lang sein können, wie man den Endspieltabellen entnehmen kann. Eine Engine, die das planlose Ausprobieren von Zugvarianten vermeiden will, braucht viel mehr Schachwissen als in den heutigen Programmen verwendet wird. Heutige Programme können auch nicht erklären, warum ein bestimmter Zug gut oder schlecht ist, denn Kausalketten werden nicht nachvollzogen, Zug und Gegenzug in einer Variante ergeben sich nur indirekt durch Rückgabe eines Stellungswertes zur Wurzel des Spielbaumes. Aber zur Beherrschung des Schachspiels ist mehr nötig, und man kann sich nur wundern, wie trotz dieser mangelhaften Methoden im Durchschnitt brauchbare Züge berechnet werden.

Zum Schluß nochmal zurück zu den Hashtabellen, die schon oben erwähnt wurden. Da diese nicht automatisch gelöscht werden, wenn man in einer schon berechneten Variante durch Drücken der Pfeiltasten der graphischen Oberfläche schrittweise vor- und zurückgeht, kann man damit nämlich den Horizonteffekt abmildern: Wenn das Programm in den Hashtabellen schon Werte aus später auftretenden Stellungen entdeckt, kann es die nutzen, um dann einen anderen, hoffentlich besseren Zug zu finden.

Das ist aber insgesamt ein schwacher Trost: Wenn es um die Wahrheit im Schach geht, kann man sich heute nur auf die Endspieltabellen verlassen, endgültige Gewissheit über die Güte eines beliebigen Schachzuges wird man durch den Alpha-Beta-Algorithmus in den meisten Fällen nicht erreichen. (*Thomas Hall* )

## Der Autor

Thomas Hall, Jahrgang 1957, studierte Informatik mit Spezialgebiet Programmiersprachen. Er interessiert sich seit seiner Jugend für Schach und ist Experte für das Morra-Gambit. Dafür hat er auch eine Computer-Eröffnungs-Bibliothek erstellt, allerdings bislang nicht veröffentlicht. In den letzten Jahren befasste er sich besonders mit Künstlicher Intelligenz und arbeitet momentan daran, Alternativen zur traditionellen Schachprogrammierung zu entwickeln.