Lars Bremer

Klipp & klar: Populäre Fehldeutungen

Moderne Schachprogramme lassen sich tief ins Hirn schauen, denn sie geben eine Menge Informationen über ihren Rechenvorgang preis. Allerdings muss man diese auch richtig interpretieren – und das scheint nicht immer ganz leicht zu sein, wie ein Rundblick über die Forenlandschaft zeigt.

„Wie schnell ist Ihr Schachprogramm und wie tief rechnet es denn so?“ Einfache Frage? „600.000 Knoten pro Sekunde und 18 Halbzüge tief“ wäre die falsche Antwort. Nein, es hat auch nichts mit magischen Wörtern wie „brute force“ und „Selektivität“ zu tun. Die Auflösung gibt es ausnahmsweise im Vorspann: Schon die Frage war falsch gestellt. Warum das so ist, kann man auch ohne Programmierkenntnisse verstehen – wenn man sich nur die Mühe macht, alte Vorurteile über Bord zu werfen.

Knoten

„Käpt‘n, was‘n das hier für ein Tau mit vielen Knoten drin?“

„Möönsch, Hein Blöd, du lernst es nicht mehr“, sprach Käpt‘n Blaubär. „Das ist doch kein gewöhnliches Tau, sondern eine Log-Leine. Die dient der Positionsbestimmung!“

Spricht ein Computerschachfreund von Knoten, meint er weder Palstek noch Englische Trompete, sondern eine (von einem Programm untersuchte) Schachposition. Für den Schachspieler ist hier alles klar. Jedes Mal, wenn das Programm intern einen Zug ausführt und so eine neue Stellung erzeugt, ist es ein Knoten mehr als vorher. Schachprogramme funktionieren aber anders als ein Gehirn. Manche erzeugen zunächst alle Züge, auch illegale. Das ist einfacher und geht darum viel schneller als das Erzeugen der legalen Züge. Beispielsweise könnte ein Programm Schachs ignorieren. Im nächsten Zug würde dann durch das Schlagen des Königs die Variante als illegal erkannt. Weil aber die Suche öfter aufgerufen wurde, hat solch ein Programm mehr Knoten berechnet als eines, das die Züge vorher auf Legalität prüft.

Die Anzeige der berechneten Knoten, wie auch immer diese Zahl programmintern zustande kommt, besagt übrigens nicht das geringste über die Anzahl der untersuchten unterschiedlichen Stellungen. Schachprogramme durchsuchen den Spielbaum iterativ. Zuerst Tiefe Eins, dann Tiefe Zwei, usw. Bei jeder neuen Iteration kommt es an vielen schon vorher mit geringerer Tiefe berechneten Stellungen vorbei und zählt diese nochmals. Wenn das Programm also eine Variante beginnend mit 1.Sf3 bis zur Iterationstiefe 18 berechnet, hat es die nach dem ersten Zug Sf3 entstehende Stellung auch mindestens 18mal gezählt, ebenso sämtliche Stellungen nach anderen ersten Zügen.

Knotenleistung

„Der Kutter ist zu langsam, Käpt‘n!“, rief ängstlich Hein Blöd.

„Wirf die Ladung über Bord, dann machen wir mehr Knoten!“ gab Käpt‘n Blaubär zurück und rettete sich, Hein und den Kutter auf diese Weise vor den grausamen Klipperklar-Piraten von Engine-Bay.

Schachprogramme sind Blaubärs Kutter sehr ähnlich – je mehr Regeln sie pro Stellung abarbeiten müssen, desto weniger Knoten berechnen sie in einer bestimmten Zeitspanne. Interpretiert man „Regeln abarbeiten“ kühn als „Schachwissen besitzen“, gelangt man schnell zum vielleicht verbreitetsten Irrtum der Computerschach-Geschichte: Ein Programm, das wenig Knoten/Sekunde berechnet, ist klüger als eines, das viele Knoten/Sekunde berechnet. Viele Knoten/s = schnell und dumm, wenig Knoten/s = langsam und „wissensbasiert“, so der funkensprühende mentale Sicherungsausfall.

Es handelt sich hier aber um einen unzulässigen Umkehrschluss. Freilich braucht zusätzlich hinzugefügter Programmcode Zeit, bis er abgearbeitet ist. Erweiterungen in der Bewertungsfunktion schlagen sich deshalb tatsächlich in weniger angezeigten Knoten/Sekunde nieder. Das bedeuten aber keinesfalls, dass jede Reduzierung der Knotenzahl auf zusätzliches Wissen zurückzuführen wäre.

Grundsätzlich drückt jeder Programmcode, der während der Suche viele tausend Male abgearbeitet wird, die Anzahl der Knoten pro Sekunde. Die Verschwendung nur einer einzigen Mikrosekunde kostet an einer Stelle, die pro ausgeführtem Partiezug eine Million Mal aufgerufen wird, 1 Sekunde Bedenkzeit. Und eine Million Knoten berechnen heutzutage doch die meisten Programme schon im Blitz.

Als Beispiel mag eine grundlegende Technik wie der Killer-Algorithmus dienen: Man stelle sich tief im Suchbaum eine einzügig hängende weiße Dame vor. Weiß macht einen Zug, der die Tante nicht in Sicherheit bringt und Schwarz schlägt sie mit einem Bauern. Dieser Bauernzug, der eine ganze Dame einheimst, wird zum Killerzug erklärt. In der weiteren Berechnung probiert Schwarz nun auf jeden anderen weißen Zug zuerst den Bauernzug, sodass sofort auffällt, wenn die Dame noch einsteht. Das Programm kann auf diese Weise viel Rechenzeit sparen, weil es in solchen Varianten die Suche abbrechen kann. Ohne den Killerzug wären vielleicht zunächst schwarze Züge untersucht worden, die die Dame nicht schlagen, und erst nach der vollständigen Berechnung dieser nutzlosen Züge hätte das Programm den Damengewinn bemerkt. Der Killer-Algorithmus hat mit Schachwissen nicht das Geringste zu tun, er funktioniert genau so auch in Dame- oder Reversiprogrammen.

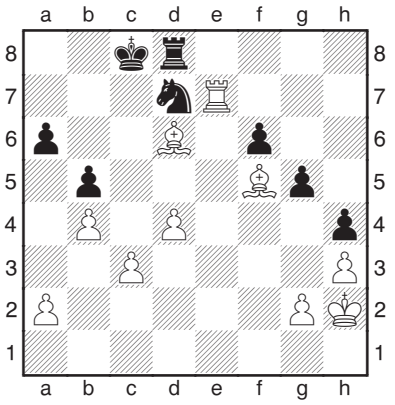
Angenommen, von zwei ansonsten identischen Programmen verwendet A diese Killer-Technik und B nicht. B berechnet dann mehr Knoten pro Sekunde, weil A Zeit braucht, sich den Zug zu merken und nach vorn zu sortieren. Trotzdem spielt A stärker als B, weil es *pro Halbzug Suchtiefe weniger Knoten benötigt* (es kann ja viele Varianten eher abbrechen) und demzufolge tiefer in den Suchbaum eindringen kann.

Dieser Effekt tritt bei jeder Technik auf, welche die Suche optimiert – der Code für den neuen und genialen Trick zur Baumbeschneidung muss eben erst einmal abgearbeitet werden und das kostet Zeit resp. Knoten/s.

Suchtiefe

„Vorsicht, Käpt'n, da vorn ist eine Untiefe!" brüllte Hein Blöd. Blaubär blieb ganz ruhig. „Keine Bange, Hein, wir segeln nur eine Abkürzung. Das da vorn sind nur die berüchtigten Stromschnellen von Abschneid, wir kommen aber gleich in den Extension-Strom, da haben wir wieder jede Menge Wasser unter'm Kiel."

Die ersten Schachprogramme, zum Beispiel die Turing-Engine, durchsuchten den Spielbaum bis zu einer bestimmten Tiefe. Jeder noch so unbedeutende Zweig wurde berechnet, jede Fortsetzung geprüft. Die Erfindung des Alpha-Beta-Verfahrens änderte daran zunächst wenig. Zwar musste ein Programm nun nicht mehr jede Variante durchsuchen, der beste Zug aber blieb derselbe. Innerhalb der Suchtiefe (gern „Brute-force-Sockel" genannt) spielte das Programm absolut göttlich und übersah nichts. Außerhalb dieser Suchtiefe aber lag ein unentdecktes Land, von dem jedes Programm nur soviel wusste wie Kermit der Frosch vom Gefühlsleben blonder Stoffschweinchen. Die saubere Trennung zwischen Perfektion und Blindheit an den Blättern des Suchbaums, dem Horizont, führte zu lustigen Effekten; hier eine Stellung von 1983:



Eine aussichtslose Situation für das schwarz spielende Programm – egal, was Schwarz anstellt, der Springer muss auch noch über die Klinge springen. Aber vielleicht kann man das Pferdchen retten, dachte der Computer und rechnete los. Tatsächlich, man lenkt den Gegner einfach mit Bauern ab: 1...g4 2.Lxg4 f5 3.Lxf5 a5 4.bxa5 b4 5.cxb4, und hier nun gab es keine Ablenkungsopfer mehr. Der Springer geht trotzdem verloren. Das Programm schob den endgültigen Verlust der Figur über den Suchhorizont hinaus – ganze vier Bauern kostete das, bevor der Springer schlussendlich doch verloren ging.

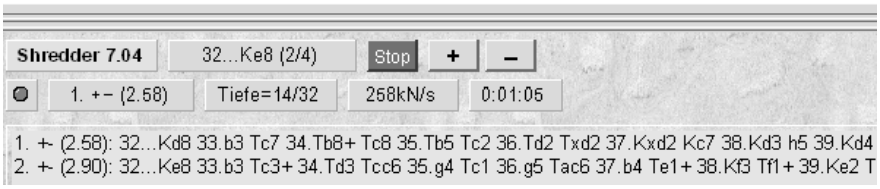


Abb. 1: Bewertung, Hauptvarianten, Suchtiefe, Knotenanzeige – Schachprogramme geben jede Menge Informationen aus, die man fast alle fehlinterpretieren kann.

Um diesen Horizonteffekt abzumildern, dachten sich die Programmierer vielerlei Tricks aus. Moderne Schachprogramme nehmen die angezeigte Brute-Force-Rechentiefe nicht mehr allzu ernst, sondern berechnen Varianten oftmals weniger tief, andere dafür umso tiefer. Ein veröffentlichtes Verfahren zur Vorwärtsabschneidung ist die Null-Move-Technik. Dabei wird davon ausgegangen, dass es ein Vorteil sein müsste, zweimal hintereinander ziehen zu können. Die Suche führt also für eine Seite einen Nullzug aus, der die Stellung nicht verändert, *reduziert die maximale Suchtiefe* und gibt das Zugrecht an die Gegenseite ab. Gelingt es dieser nicht, ihre Bewertung (innerhalb der verkürzten Suche!) damit zu verbessern, taugt mit großer Wahrscheinlichkeit die ganze Variante nichts und es erfolgt ein Suchabbruch. Allerdings ist die Annahme, es sei von Vorteil, zweimal hintereinander ziehen zu dürfen, nicht in jedem Falle richtig, weil besonders im Endspiel manchmal Situationen auftreten, in denen es nachteilig ist, am Zug zu sein.

Auch bei anderen Vorwärtsabschneidungen kann es vorkommen, dass ab und zu ein wichtiger Zug erstmal durch den Rost fällt und von dem Programm erst viel später gefunden wird, als die angezeigte nominelle Suchtiefe vermuten ließe.

Lassen die Programmierer ihre Schöpfungen also eine bestimmte Suchtiefe anzeigen, verschweigen sie uns schamhaft, dass es sich eigentlich gar nicht wirklich um eine Suchtiefe handelt, sondern nur um eine interne Variable, welche die vom Programm durchlaufenen Iterationen zählt. Stolz präsentiert man uns dafür die Vertiefungen, Varianten, die ein Programm weit tiefer als normal untersucht. Übrigens reicht schon ein kurzer Blick durch die Mathebrille, um zu erkennen, wie viel die Programme vom Suchbaum abschneiden, denn die höchstmögliche Effizienz eines reinen Alpha-Beta-Algorithmus wird locker übertroffen.

Wenn der neue Fritz oder Shredder also anzeigt, „Tiefe = 18/43", so heißt das keineswegs, dass sämtliche Varianten bis zu einer Tiefe von 18 Halbzügen untersucht worden wären. Dass manche Programme auch eine ganz vom üblichen abweichende Technik erfolgreich umsetzen können, zeigt der vielfache Computerschach-Weltmeister Junior, dessen Iterationen kaum etwas mit der tatsächlichen Suchtiefe zu tun haben. Auch „The King" liefert eine von der Fritz- und Shredder-Norm abweichende Suchtiefenanzeige.

Über eine neue Programmversion zu sagen „sucht drei Halbzüge tiefer als der Vorgänger" ist äquivalent zu „im Unterschied zum Vorgängermodell hat der neue Passat eine leicht geschwungene Kofferklappe", keineswegs aber zu „der neue Passat hat 50 PS mehr als der Vorgänger". Es wäre für jeden Schachprogrammierer einfach, eine Version zu erstellen, die 100 Halbzüge tief sucht. Nur würde ein solcher Tiefenbohrer leider nicht vernünftig Schach spielen, weil zu viele wichtige Varianten aus dem Suchbaum geschnitten wurden.

Fazit

„Käpt'n, ich glaub' Ihnen kein Wort." – „Aber Hein, ich sage die lautere Wahrheit!"

Wie schnell ist mein Programm denn jetzt wirklich? Wer das für wirklich wichtig hält, muss schon Lösezeiten messen. Ansonsten gilt: Wenn Ihnen jemand von größerer Suchtiefe erzählt, von schnelleren und langsameren Programmen oder von ach so viel Wissenszuwachs einer neuen Programmversion, der deutlich an der geringeren Knotenleistung zu erkennen sei – schenken Sie ihm genauso viel Vertrauen wie einem blauen Bären mit Kapitänsmütze.