Über kurz oder lang ...

Die Frage, ob Engine-Matches mit kurzer Bedenkzeit Schlüsse auf die Ergebnisse bei längerer Bedenkzeit erlauben, ist so alt wie das Computerschach und wird seitdem höchst kontrovers diskutiert. Gerhard Sonnabend, der Autor dieses Artikels, beschäftigt sich bereits einige Jahre mit diesem Thema und stellt sein laufendes Projekt zu diesem ewig aktuellen Thema vor.

Über kurz oder lang ...

Blitz vs. Aktivschach

Seit einiger Zeit führe ich zwei neue Ratinglisten, um die Spielstärke der Top-Programme zu ermitteln und um die Unterschiede zwischen Blitz- und Aktivschachratings unter sonst identischen Bedingungen zu ermitteln. Die Programme haben dabei für jeweils 40 Züge 15 Minuten (Aktivschach) bzw. 3 Minuten (Blitz) Zeit, die Bedenkzeiten unterscheiden sich also um den Faktor 5.

Die Programme treten in einem Rundenturnier jeder gegen jeden unter der ChessBase-Shredder-Oberfläche auf einem Pentium 4 mit 2 GHz und je 64 MByte Hash gegeneinander an, wobei sie 25 Vorgabestellungen mit beiden Farben spielen, insgesamt also 50 Partien pro Match. Permanent Brain wurde deaktiviert, der PC nach jedem Durchgang neu gestartet und eventuell entstandene Lerndateien landen nach jedem Match auf dem Müll.

Derzeit (nach jeweils 3300 Partien) ergibt sich folgendes Bild:

Level 40/15 + 40/15 + ...								
	Engine	Rating	Punkte	aus	+	=	-	Ø Gegner
1	Rybka 1.0 32-bit	2684	346.0	550	246	200	104	2592
2	Fruit 2.2	2655	322.0	550	231	182	137	2595
3	Fritz 9	2651	318.5	550	241	155	154	2595
3	Hiarcs 10	2651	318.5	550	215	207	128	2595
5	Shredder 9 UCI	2640	309.5	550	231	157	162	2596
6	Loop List Release	2577	255.5	550	166	179	205	2602
7	Spike 1.0a Mainz	2576	254.0	550	148	212	190	2602
8	Chess Tiger 15.0	2573	251.5	550	150	203	197	2602
9	SmarThink 1.00	2558	239.0	550	156	166	228	2604
10	Gandalf 6.0	2556	237.0	550	146	182	222	2604
11	Ruffian 2.1.0	2547	229.5	550	142	175	233	2605
12	Ktulu 7.1	2534	219.0	550	125	188	237	2606

Level 40/3 + 40/3 + ...								
	Engine	Rating	Punkte	aus	+	=	-	Ø Gegner
1	Hiarcs 10	2686	347.5	550	260	175	115	2592
2	Fritz 9	2682	344.5	550	268	153	129	2593
3	Rybka 1.0 32-bit	2680	342.5	550	263	159	128	2593
4	Fruit 2.2	2666	331.0	550	242	178	130	2594
5	Shredder 9 UCI	2648	316.0	550	241	150	159	2596
6	Chess Tiger 15.0	2633	303.5	550	205	197	148	2597
7	Loop List Release	2570	249.5	550	153	193	204	2603
8	SmarThink 1.00	2551	233.0	550	146	174	230	2605
9	Spike 1.0a Mainz	2540	223.5	550	129	189	232	2606
10	Ktulu 7.1	2525	211.0	550	137	148	265	2607
11	Ruffian 2.1.0	2518	205.5	550	122	167	261	2608
12	Gandalf 6.0	2501	192.5	550	122	141	287	2609

Man erkennt bereits auf den ersten Blick deutliche Abweichungen. Ein kleiner Auszug aus den Einzelmatches zeigt besonders krasse Unterschiede (alle Matchresultate finden sich unter www.pcschach.de).

Ergebnis Nummer 1 = Level 40/15+40/15+ ..., Nummer 2 = Level 40/3+40/3+ ...

Shredder 9 vs Fruit 2.2	28.5 - 21.5	+23	=11	-16
Shredder 9 vs Fruit 2.2	17.5 - 32.5	+12	=11	-27

Fruit 2.2 vs Gandalf 6	33.0 - 17.0	+25	=16	-09
Fruit 2.2 vs Gandalf 6	39.5 - 10.5	+34	=11	-05
Fruit 2.2 vs Hiarcs 10	27.0 - 23.0	+17	=20	-13
Fruit 2.2 vs Hiarcs 10	21.5 - 28.5	+12	=19	-19
Fritz 9 vs Hiarcs 10	20.0 - 30.0	+10	=20	-20
Fritz 9 vs Hiarcs 10	26.0 - 24.0	+19	=14	-17
Spike 1.0a Mainz vs Chess Tiger	29.5 - 20.5	+19	=21	-10
Spike 1.0a Mainz vs Chess Tiger	18.5 - 31.5	+08	=21	-21
SmarThink 1.00 vs Fritz 9	16.0 - 34.0	+13	=06	-31
SmarThink 1.00 vs Fritz 9	23.0 - 27.0	+15	=16	-19

Niederlagen wandeln sich also gelegentlich in Siege und umgekehrt. Zuerst ist natürlich anzumerken, dass 50 Partien pro Match nicht besonders viel sind. Allerdings sind die Unterschiede auch insgesamt (hier bisher immerhin 2 x 550 Partien pro Engine) deutlich spürbar. Warum das so ist, weiß keiner so ganz genau; Gründe könnten sein:

Die Engine teilen sich auf den zwei genannten Leveln die Bedenkzeiten total unterschiedlich ein. Beim Blitz kann man nach Betrachten bereits weniger Partien feststellen, dass sehr oft bereits im Mittelspiel, spätestens jedoch im späten Mittelspiel oder frühen Endspiel Zeitnot ein beherrschendes Thema ist. Auch Zeitvorgaben wie 5 Minuten plus 5 Sekunden helfen dem nicht spürbar ab!

Einige Programme schalten bei zu geringer Bedenkzeitvorgabe, mache sogar bei zu geringer Restbedenkzeit, gewisse Programmteile ab (oder auch zu!) oder verändern den Einsatz der Suchverfahren.

Viele Zugkandidaten werden, wenn das Programm den „Ziehe jetzt“-Befehl erhält, noch sehr ähnlich bewertet. Auch aus diesem Grund sind viele Züge zufälliger Natur, weil das Programm zwischen vielen scheinbar gleichwertigen Alternativen wählen kann, die einfach noch nicht tief genug untersucht wurden, um deutliche Bewertungsunterschiede zu entdecken. Viel hängt dann davon ab, welcher Zug zuerst betrachtet wurde – es entstehen unterschiedliche Suchbäume, Abschneidungen und Vertiefungen springen an (oder eben nicht), die Alpha-Beta-Schranken sind andere, der Hashinhalt differiert, kurz: Der entstehende Suchbaum ist ein anderer, als wenn ein scheinbar gleichwertiger Alternativzug zuerst untersucht worden wäre. Wer einmal versucht hat, Matches unter sonst identischen Bedingungen mit sehr kurzer Bedenkzeit zu reproduzieren, der wird wissen, was ich meine.

Um Hardware und Rahmenbedingungen zu überprüfen, habe ich zusätzlich einen Plytest laufen lassen. Die Gegner waren Spike und Chess Tiger sowie Fruit 2.2 und Shredder 9, welche mit einer festen Suchtiefe von neun und sieben Halbzügen je dreimal über den Parcours gejagt wurden. Laut LoopList-Programmierer Fritz Reul könnten hier die Resultate reproduzierbar sein, falls gewisse Bedingungen wie identischer Hashinhalt usw. erfüllt sind. Die Ergebnisse zeigen das ziemlich gut:

Spike 1.0a Mainz vs Chess Tiger 15.0 (feste Tiefe = 9)	
1. Durchgang	23.5 - 26.5
1. Wiederholung	23.5 - 26.5
2. Wiederholung	23.5 - 26.5

Fruit 2.2 vs Shredder 9 (feste Tiefe = 7)	
1. Durchgang	29.5 - 20.5
1. Wiederholung	29.5 - 20.5
2. Wiederholung	29.5 - 20.5

Ein weiterer Punkt könnten die unterschiedlichen Aktivitäten im Hintergrund solcher Matches sein, also die unterschiedlichen Tasks, die unter Windows laufen. Auf der Testmaschine ist es aber so, dass die Programme meist 100 Prozent (und niemals unter 99 Prozent) des Prozessors nutzen dürfen. Das liegt daran, dass es sich um einen reinen Schach-PC handelt, auf welchem sich nicht einmal eine Textverarbeitung, ein Internetzugang, Virenscanner und/oder Firewall oder sonstige „Störer“ befinden. Auch der automatische „Windows-Update-Erinnerer“ ist gelöscht, der Ordner Autostart leer, überflüssige Dienste laufen nicht. Jedoch kann man diese Einflüsse mit Sicherheit nicht vollkommen ausschließen. In Partien mit sehr kurzer Bedenkzeit wirkt sich jede noch so kleine Störung logischerweise stärker aus, als bei langer Bedenkzeit, denn die durch die Störung „fehlgeleitete“ Bedenkzeit ist prozentual umso höher, je niedriger

die Spielstufe gewählt wurde!

Bei kurzer Bedenkzeit sind Matchresultate also weniger gut reproduzierbar als bei langer Bedenkzeit, soviel scheint klar. Während im Aktivschach schon relativ kurze Matches ziemlich gleiche Ergebnisse liefern, sind auf Blitzstufe hierfür weit mehr Partien nötig. Die spannende Frage lautet, wo der Break-even liegt, bei welcher Zeitvorgabe das „wahre“ Ergebnis angenähert auftritt und dann auch bei weiterer Steigerung der Bedenkzeit stabil bleibt. Zu diesem Zwecke laufen gerade lange Testserien mit jeweils verdoppelter Bedenkzeit. Die Auswertung dieser Testserien soll Thema der Fortsetzung dieses Artikels in einer der nächsten Ausgaben von CSS Online sein. Momentan liegen folgende Ergebnisse für das erste Engine-Paar, Fruit 2.2 und Hiarcs 10, vor, die sich in 70 Vorgabestellungen mit 140 Partien pro Match bei verschiedenen Bedenkzeiten bekriegen:

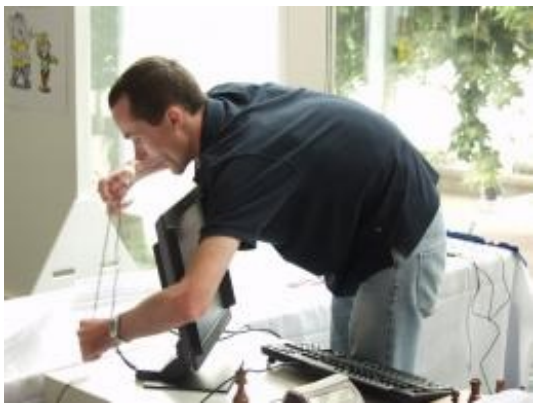
Level 1+1	Fruit 2.2 vs Hiarcs 10	53.5 - 86.5	+36	=35	-69
Level 2+1	Fruit 2.2 vs Hiarcs 10	68.5 - 71.5	+53	=31	-56
Level 4+1	Fruit 2.2 vs Hiarcs 10	64.0 - 76.0	+44	=40	-56
Level 8+1	Fruit 2.2 vs Hiarcs 10	69.5 - 70.5	+43	=53	-44

Über kurz oder lang ...

Programmierer über Bedenkzeit

Wer sollte besser Bescheid wissen über unterschiedliches Verhalten eines Programms bei verschiedenen Bedenkzeiten als der Programmierer? CSS Online hat daher einige davon befragt, Fritz Reul, Autor von Loop List, Shredder-Schöpfer Stefan Meyer-Kahlen und Chrilly Donninger, der den momentan stärksten Schachcomputer *Hydra* programmiert hat.

CSS Online: *Warum unterscheiden sich Matchergebnisse in Abhängigkeit von der gewählten Spielstufe teilweise so deutlich ? Hast Du dies auch schon beobachtet?*


Fritz Reul bei der Chess960-WM

Fritz Reul: Eigentlich unterscheiden sich die Matchergebnisse nicht so drastisch. Es sei denn es werden nur wenige Partien gespielt (<100). Mit zunehmender Anzahl nähern sich die Ergebnisse wieder sehr stark an! Die Unterschiede in einzelnen Partien/Serien liegen eben an diesen Schwelleneigenschaften in den Zeitkontrollen.

SMK: Ein großer Anteil der beobachteten Phänomene lässt sich sicher mit der geringen Partienanzahl erklären. Es ist leider so, dass man sehr viele Partien braucht, bis man fundierte Aussagen manchen kann. Es ist aber sicher auch der Fall, dass einige Engines mit mehr Bedenkzeit oder mehr Rechenpower relativ besser werden.

Chrilly Donninger: Computer Programme sind allgemein "Chaotische Systeme". Der Schmetterling schlägt in China seine Flügel zusammen und alles ist anders. Manchmal verändert man sehr grosse Teile und alles bleibt beim Alten, manchmal verändert man nur eine Kleinigkeit, und das Programm spielt wie ausgewechselt. Bei der Spielstufe gibt es Sprünge. Angenommen, das Programm rechnet bei 30 Sekunden pro Zug bis Tiefe 14. Dann wird es bei 35 Sekunden pro Zug wahrscheinlich auch auf Tiefe 14 rechnen. Aber irgendwann erreicht es dann zumindest bei ein paar (wesentlichen) Zügen Tiefe 15. Bei Programm A tritt der Sprung z.B. bei 40, bei B bei 50 Sekunden auf. Dann ist das Ergebnis bei 30 Sekunden und 40 Sekunden ganz anders.

Ferner kann man unterschiedlich optimieren. Z.B. ist es für kurze Bedenkzeiten besser, wenn man stark erweitert, für längere Bedenkzeiten sollte man die Erweiterungen eher reduzieren.

CSS Online: *Weshalb ist es so schwer Matchresultate zu reproduzieren? Mir ist dies vor allem bei sehr niedrigen Bedenkzeitvorgaben (=Blitz) aufgefallen.*

Fritz Reul: Nur wenn die Zufallszahlen für die Hashcodierung beim Neustart der Engine diesselben sind, kann eine Partie reproduzierbar sein. Bei List ist dies der Fall, Fruit geht sogar noch einen Schritt weiter und speichert die Zufallszahlen in einer Tabelle ab. Wie es bei H10 und F9 ist, keine Ahnung.

Durch Schwelleneffekte in der Zeitkontrolle ist es kaum möglich, Partien zu reproduzieren. Kennst Du Ply-Tests? Hier lassen sich Partien immer reproduzieren, wenn obiges Kriterium erfüllt ist. Meiner Ansicht nach hängt das NUR an der Zeitkontrolle bzw. dem Zeitverhalten.

SMK: Pondern, das Nichtlöschen des Hash und Lernen führen dazu, dass im Prinzip keine Partie mehr völlig reproduzierbar ist. Bei parallelen Programmen wird die Sache noch viel schlimmer. Das macht die Sache nicht einfacher, wenn irgendwo Probleme oder Fehler auftauchen, die man debuggen muss. Bei kurzen Bedenkzeiten ist der Quotient Anzahl der Wechsel des besten Zuges pro Sekunde viel größer als bei längeren Bedenkzeiten. Ich denke, dass macht das Reproduzieren hier auch schwieriger, da bei einer leichten Änderung des Hardwaretimings von 0.1 Sekunden die Wahrscheinlichkeit größer ist, dass der beste Zug wechselt. Ein nichtgelöschter Hash hat bei kurzen Bedenkzeiten auch einen viel größeren Einfluss als bei langen.

Chrilly Donninger: Abgesehen vom obigen Effekt sollte man die Rolle der Statistik nicht unterschätzen. Selbst bei 100 Partien hatte man noch eine ordentliche Streuung. Das ist z.B. ein Problem, wenn man verschiedene Versionen testet. Irgendwann bekommt man schon rein aus statistischen Gründen eine "Super-Version". Wenn man diese Super-Version dann noch einmal 100 Spiele machen läßt, schneidet sie dann wieder ganz normal ab.

CSS Online: *Schaltet Deine Engine Programmteile in Abhängigkeit von der gewählten Spielstufe und/oder der noch zur Verfügung stehenden Zeit zu oder ab? Wie geht Dein Programm mit dem Faktor Zeit (Restzeit) um?*

Fritz Reul: Nein, nichts wird angeschaltet. Blitzbeispiel mit 900 Sekunden Basiszeit und 10 Sekunden Zeitaufschlag. Man nehme z.B. 90 Prozent der Basiszeit weniger zwei oder drei Sekunden Puffer (falls die GUI-Uhr nicht synchron zu Engine Uhr ist). Zu dieser Zeit t addiert man das 20- bis 30-fache des Zeitaufschlags. Diese Zeit teil man wiederum durch das 20- oder 30-fache, das wars.

time=((900*0.9)-2+(20-1)*10)/20

SMK: Das passiert in der Regel nicht über die verbleibende Bedenkzeit oder die Rechenzeit,

Chrilly Donninger

sondern über die Suchtiefe, z.B. mache dies und das nur bei Resttiefe größer X. Dieser Effekt ergibt sich somit automatisch.

Chrilly Donninger: Nein. Bleibt immer gleich. Das erscheint mir zu kompliziert, und bei Hydra bringt das auch nichts. Wenn man in einem Hardware-Programm etwas ausschaltet, dauert es länger als wenn man es immer mitlaufen laesst. Der Zeitalgorithmus ist ziemlich kompliziert. Im wesentlichen ist es:

Zielzeit = RestZeit / (Züge zur Zeitkontrolle + 2).

Das wird aber modifiziert. Wenn es z.B. viele legale Züge in der Ausgangsstellung gibt, wird der Wert erhöht. Die Zielzeit wird auch nach jeder Iteration modifiziert. Z.B. gibt es eine "Easy-Move" Erkennung. Da wird die Zeit nach jeder Iteration halbiert (die "Easy-Move" Erkennung muss nicht in jeder Iteration "zuschlagen", da Such-Statistiken verwendet werden). Ferner wird versucht, schwierige Stellungen zu erkennen. Da gibts dann einen extra Schluck aus der Zeitflasche. Und noch den Panik-Modus, falls das Programm ein tiefes Fail-Low hat. Da wird die Zeit sehr stark erhöht. Meistens ist das Unglück aber schon vorher passiert (Hydra hat aber auch selten Fail-Lows).

Ferner hat auch das automatische Buch einen Einfluss auf das Zeitverhalten. Wir haben nur ein sehr kurzes normales Eröffnungsbuch (ca. 10 Züge). Danach gibt es ein automatisch erstelltes Buch. Wenn Hydra darin einen häufig gespielten Zug findet und diesen Zug auch selber spielen will, dann wird kürzer gerechnet. Die Einsparung von Zeit ist die Hauptfunktion des automatischen Buches. Es wird gleichzeitig vom Programm überprüft, dass der Buchzug kein totaler Schrott ist (beim Buch von GM Lutz steht mit ziemlicher Sicherheit kein Schrott drinnen, das automatische Buch entstand aber noch zu Erdogans Buchzeiten. Das Erdo-Buch hat eine ziemlich hohe Schrott-Rate).

CSS Online: *Einige Tester von Schachprogrammen vertreten die Meinung, dass eine große Anzahl an Blitzpartien ausreiche, um sehr gut auf die Stärkenverhältnisse bei längeren Bedenkzeiten schliessen zu können.*

Fritz Reul: Ja, das sollte zutreffen. Zumindest verlasse ich mich auf diese Hypothese, seit ich List entwickle. Aus der Not heraus bleibt den Entwicklern keine andere Möglichkeit als zu Blitzen! Wer hat schon die Ressourcen für höhere BZ :-)


Stefan Meyer-Kahlen

SMK: Wenn eine Engine +50 Elo im Blitz gegen eine andere holt, dann wird es für die andere Engine schwer, bei Turnierbedenkzeiten Ausgleich zu kriegen. Wenn man sich mal überlegt, dass eine Verdopplung der Rechenzeit +50 Elo bringt und eine Engine bei 15 Minuten/Partie eine andere mit +50 Elo schlägt, dann ist es wohl nicht möglich, dass bei 30 Minuten/Partie schon Gleichstand herrscht. Die eine Engine hätte sich dann 50 Elo gesteigert, die andere müsste dann bei 15 Minuten und 30 Minuten genau gleich gut spielen, eher unwahrscheinlich. Es gibt aber wie gesagt Engines, die mehr von mehr Bedenkzeit profitieren, der Unterschied hier ist aber nicht so krass.

Chrilly Donninger: Das ist bestensfalls ein erster Eindruck. Man muss immer das testen, was man messen will. Wenn man die Stärke im Blitz messen will, muss man Blitzpartien spielen, wenn man die Stärke unter Turnierbedingungen messen will, dann muss man (auf möglichst starker Hardware) unter Turnierbedingungen spielen. Nach meinen Erfahrungen darf man höchstens um den Faktor 2 unterschiedliche Einstellungen nehmen, d.h. statt 40 Züge in zwei Stunden, 40 Züge in einer Stunde. Dann kann es aber immer noch zu dem eingangs beschriebenen Kipp-Effekt kommen. Aber den hat man auch, wenn man auf besserer oder schlechterer Hardware spielt.

Ganz allgemein ist das Messen von Verbesserungen oder Verschlechterungen das Problem schlechthin bei der Entwicklung. 100 Elo Unterschied merkt man sofort, da braucht man auch nicht viel zu testen, aber 10 Elo sind kaum zu messen. Die meisten Verbesserungen sind (leider) im 10 Elo Bereich.

CSS Online: *Vielen Dank!*

(Gerhard Sonnabend)