Computerschach: Neue Testaufgaben für Programme

5. März 2017 Computerschach, Schach, Testaufgaben
Walter Eigenmann

Der «Eigenmann Rapid Engine Test» (ERET)

von Walter Eigenmann

Inhaltsverzeichnis des Beitrages [ausblenden]

NAMENSGEBER DES GLAREAN MAGAZINS

... ist der Schweizer Musiker, Dichter, Mathematiker und Humanist Glarean

SUCHEN IM GLAREAN MAGAZIN

Suchen ...

FOLGEN SIE UNS

...und erhalten Sie bei jedem neuen Beitrag eine Mail-Nachricht:

Der sog. **Eigenmann Rapid Engine Test** (ERET) ist eine neue Sammlung von 111 Aufgaben für Schach-Programme. Er wurde konzipiert, um schnell einen ersten Eindruck von der Spielstärke einer neuen Engine ausmachen zu können. Die Computerschach-Anwenderschaft erhält mit diesem ERET erstmals eine Test-Suite an die Hand, deren Ergebnisse innert zehn Minuten eine grobe, aber recht verlässliche Einschätzung eines (neuen) Programms erlauben.

Die 111 Stellungen bzw. ihre Hauptvarianten können hier nachgespielt und als PGN-Datei heruntergeladen werden. Downloadbar ist ausserdem der ERET-Test im CBH-Format (für Chessbase-Software) sowie als EPD-Datei für den Import in diverse Schach-GUI's.

Die Vorzüge des ERET gegenüber älteren Sammlungen sind namentlich:

- Grosse Bandbreite der Schachmotivik
- Eindeutigkeit der Lösungen
- Ausgewogenheit der Partiephasen
- Mittlerer bis hoher Schwierigkeitsgrad
- Keine Auswertungsformeln
- Schnelle Programm-Resultate bereits nach 10 Min.
- Auch für zukünftige Engine-Generationen tauglich

DIE INHALTE

DIE NEUEN BEITRÄGE

Die Axiomatik des ER-ET-Stellungstests

Eine Aufgaben-Sammlung wie der ERET-Stellungstest für Schachprogramme basiert auf einer eigenen Axiomatik.

Diese lässt sich in den folgenden fünf Punkten zusammenfassen:

1. Die 111 Aufgaben des ERET decken einen grossen Bereich der (computer-)schachlichen Realität ab; diese *abgestimmte Kompaktheit der Zusammenstellung* ist weder durch Hinzufügungen noch Wegstreichungen antastbar.
2. Für Computerprogramme (anders als für Menschen) ist eine Schachpartie *grundsätzlich* eine Sammlung von Einzel-Aufgabenstellungen unter definierten Bedingungen – ein Stellungstest also.
3. Die Denk- bzw. Rechengeschwindigkeit beim Schachspielen ist eine *maßgebliche* Komponente der «Spielstärke».
4. Das vom Test-Autor empfohlene *Test-Setting ist ein integraler Bestandteil* des Stellungstests.
5. Unter Berücksichtigung bzw. Wahrung der Punkte 1 bis 4 garantiert der ERET keine 100%ige, aber eine weitgehende *Übereinstimmung seiner Testergebnisse mit den durchschnittlichen Resultaten des praktischen Engine-Turnierbetriebes*.

Quartett im September 2017 17. September 2017
C. Drews: Sonntags fehlst du am meisten (Roman) 16. September 2017
Internationaler Lyrik- und Kurzprosa-Wettbewerb 15. September 2017
Christopher Wood: Requiem (CD) 14. September 2017
Ausschreibung für Phantastische Literatur 12. September 2017
Das Musik-Zitat der Woche: Ursula Petrik 10. September 2017

DIE STICHWÖRT

Kategorie au

DAS ARCHIV

Wähle den M

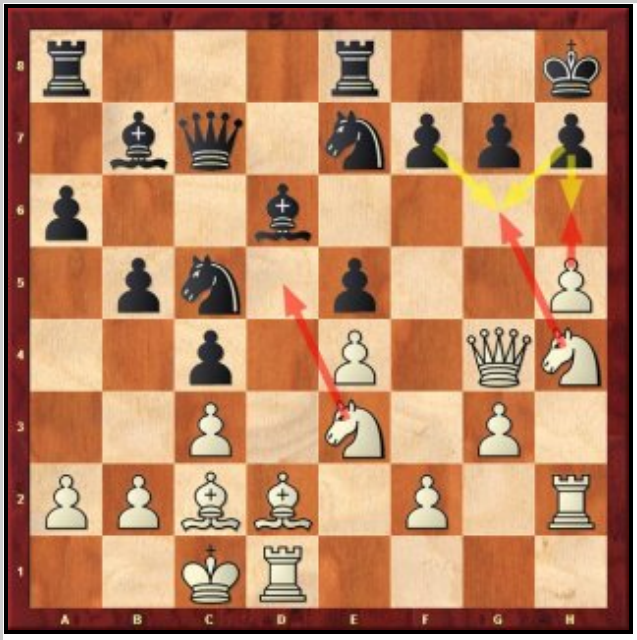
UNSERE PARTNER

Test-Setting / Anmerkungen

Die technische Durchführung des Tests ist einfach und bei den verschiedenen Schach-Interfaces wie z.B. Arena, Shredder, Aquarium oder Fritz im Grundsatz ähnlich, wobei der Autor auf aktueller Computer-Hardware (2017) das folgende Test-Setting empfiehlt:

- **Bedenkzeit:** 5-10 Sekunden pro Aufgabe
- **Prozessoren/Threads:** 1-4
- **Hash-Memory:** 256-512Mb
- **Opening-Books:** beliebig
- **Endgame-Tablebases:** beliebig
- **Extra Halbzüge:** 2-99

- Die An-



Der ERET-Schachtest für Computer-Programme deckt ein weites Spektrum an Eröffnungs-, Mittelspiel- und Endspiel-Elementen ab. Namentlich enthält er exemplarische Beispiele der folgenden taktischen und positionellen Motive (alphabetisch): Ablenkung – Abtausch – Damenopfer – Entlastung – Entwicklung – Festung – Freibauer – Initiative – Königsangriff – Königssicherheit – Läuferopfer – Linienöffnen -Mobilität – Offene Linien – Positionelles Opfer – Qualitätsopfer – Räumung – Rochadeangriff – Springeropfer – Starke Felder – Unterverwandlung – Vergif-

teter Bauer – Verteidigung – Zentralisierung – Zentrum – Zugzwang – Zwischenzug (*Bild: Die Brennpunkte des weißen Rochadangriffs in einer Fernschachpartie Copie-Patrici 1986*)

zahl Lösungen einer Engine ergibt deren wichtigstes Testergebnis; dieses erlaubt bereits einen groben Vergleich mit anderen Programmen. Um die Resultate mehrerer Engines noch zu differenzieren, empfiehlt der Autor das Interface «Fritz» ab Version 11, dessen Testergebnisse – aus der CBH- in eine PGN-Datei konvertiert – dann mit dem Freeware-Tool **EloStatTS104** von Dr. F. Schubert abgeglichen werden sollten. Diese mathematisch-statistisch fundierte Methode der Test-Auswertung ist wenn immer möglich vorzuziehen. Außerdem ist bei einer Verwendung des Testes mit «Fritz» die von EloStatTS104 mitgelieferte «Offset»-Test-File zu berücksichtigen: Mit ihm lassen sich die unterschiedlichen Reaktions- bzw. Initialisierungszeiten der Engines unter diesem Interface eruieren, womit nochmals genauere Ergebnisse generiert werden können (siehe hierzu die entspr. Readme-Datei).

Eine Alternative zum «Fritz»-GUI ist die Freeware-Oberfläche «Arena», die mit zusätzlichen Features beim Automatisierten Stellungstesten aufwartet und auch einen Output der Engine-Berechnungen bietet, allerdings auf jegliche Ranglisten-Generierung verzichtet bzw. nur die Anzahl Lösungen angibt, so dass keine weitergehende Differenzierung der Testergebnis-

se möglich ist bzw. manuell erfolgen müsste. Andere Benutzeroberflächen bieten ebenfalls nur rudimentäre Optionen bezüglich Stellungstests und sind deshalb für den ERET nur bedingt zu empfehlen.

- Moderne Rechner mit Multi-Prozessoren- bzw. -Threads-Technik neigen zu Fluktuationen in ihrer Zug-Generierung. Deren Auswirkung in der Praxis wird zwar generell sehr überschätzt, aber wer auf Nummer sicher gehen will, macht pro Engine drei bis fünf Test-Durchläufe und nimmt den Durchschnitt der jeweiligen Lösungszeiten.
- Um Software-übergreifende Vergleiche zu ermöglichen, sollten die Engines grundsätzlich mit ihren Default-Parametern getestet werden – natürlich abgesehen von Thread-, Hash- oder Tablebase-Einstellungen.
- Auf schneller Hardware reichen 5 Sekunden pro Aufgabe aus; das entspricht ungefähr der im modernen Engine-Turnier-Betrieb häufig angewendeten Bedenkzeit von 40 Zügen in 4 Minuten. Bei langsameren Rechnern empfehlen sich 7-10 Sekunden.
- Bei der Generierung von Ranglisten mit dem ERET sollten nicht die absoluten Zahlen-Ergebnisse, sondern vielmehr die Engine-Relationen beachtet werden – so wie bei den Rankings der verschiedenen bekannten Engine-Turnieren auch. Diese können bekanntlich (je nach Turnier-Settings und Berechnungsgrundlage) überra-

schend unterschiedlich ausfallen – siehe diesbezüglich auch die einschlägig bekannten Ranglisten wie z.B.u.a. hier: CEGT, CCRL, FCP, OHCR, SPCC, FGRL. Hier ein aktueller Vergleich von fünf dieser häufig zitierten Ranglisten im Internet mit den ERET-Ergebnissen.

- Die einzelnen Stellungen hat der Autor hauptsächlich mit den drei Programmen *Deep-Shredder 11/12*, *Critter 1.6* und *RybkaWinFinder 2.2* im Hinblick auf ihre taktische Korrektheit untersucht. Ansonsten hat er auf weiteres Einzeltesten bewusst und konsequent verzichtet, um den Test möglichst objektiv und nicht irgendwie «geeicht» auf bestimmte Programme gestalten zu können. (Über entsprechende Resultat-Meldungen aus der Leser- bzw. Anwenderschaft per E-Mail würde sich der Autor also sehr freuen!).
- Trotz der relativ kurzen Zeit-Vorgabe von 5 Sekunden/Stellung sind die ERET-Aufgaben keineswegs trivial. Viele der Stellungen dürften sogar ganz besondere Knacknüsse auch für heutige Engines sein. Die umfangreichen persönlichen Analysen mithilfe der obengenannten Programme legen den Schluss nahe, dass dieser Test eher im oberen Schwierigkeitsbereich angesiedelt ist. Der Autor ist deshalb zuversichtlich, dass der ERET auch noch in fünf oder zehn Jahren interessant sein wird...

Aktuelle ERET-Ranglis-

te (Mai 2017) – Generiert mit EloStatTS104

Program Elo +/- Matches Score Av.Op. S.Pos. MST1 MST2

1 AsmFish 2017-03-15 (4CPU) : 3421 3 8204 68.5 % 3286 72/111 2.0s 3.0s

2 Stockfish 170417 (4CPU) : 3414 3 7923 67.6 % 3286 67/111 1.9s 3.1s

3 Komodo 10.4 (4CPU) : 3411 3 8074 67.2 % 3286 68/111 2.0s 3.2s

4 Stockfish 8 (4CPU) : 3409 3 7946 67.0 % 3286 64/111 1.8s 3.2s

5 Brainfish 190317 (4CPU) : 3408 3 8080 66.8 % 3286 65/111 1.9s 3.2s

6 Houdini 5.01 (4CPU) : 3404 4 7617 66.2 % 3288 62/111 1.9s 3.2s

7 Komodo 10.3 (4CPU) : 3397 4 7774 65.3 % 3287 60/111 1.9s 3.3s

8 Stockfish 8 (1CPU) : 3390 4 7188 64.3 % 3287 57/111 1.9s 3.4s

9 Houdini 5.01 (1CPU) : 3384 4 6732 63.4 % 3289 54/111 2.0s 3.4s

10 Fizbo 1.9 (4CPU) : 3382 4 6725 62.8 % 3291 50/111 1.7s 3.5s

11 Komodo 10.4 (1CPU) : 3382 4 7094 63.2 % 3288 56/111 2.2s 3.6s

12 Komodo 10.3 (1CPU) : 3379 4 6718 62.7 % 3289 52/111 1.9s 3.5s

13 McBrain 2.1 (1CPU) : 3373 4 6558 61.9 % 3289 51/111 2.1s 3.6s

14 Stockfish 7 (1CPU) : 3372 4 6826 61.8 % 3289 53/111 2.3s 3.7s

15 Komodo 10 (1CPU) : 3367 4 6220 60.7 % 3292 48/111
2.1s 3.7s
16 DeepShredder 13 (4CPU) : 3362 4 6205 60.1 % 3291
46/111 2.0s 3.7s
17 Houdini 4 (4CPU) : 3361 4 6278 59.7 % 3292 45/111
1.9s 3.7s
18 Komodo 9.42 (1CPU) : 3359 4 6127 59.7 % 3291 45/111
2.1s 3.8s
19 Andscacs 0.90 (4CPU) : 3358 4 6044 59.4 % 3292 42/111
1.7s 3.7s
20 Stockfish 6 (1CPU) : 3353 4 5734 58.7 % 3292 42/111
1.9s 3.8s
21 Komodo 9.2 (1CPU) : 3352 4 5981 58.6 % 3292 44/111
2.2s 3.9s
22 Andscacs 0.89 (4CPU) : 3349 5 5872 57.9 % 3293 43/111
2.1s 3.8s
23 Fritz 15 (4CPU) : 3349 4 5935 57.7 % 3294 41/111
2.0s 3.9s
24 Komodo 9 (1CPU) : 3348 4 5827 57.9 % 3292 40/111
1.9s 3.8s
25 Fizbo 1.9 (1CPU) : 3346 5 6014 57.3 % 3295 42/111
2.2s 3.9s
26 Critter 1.6a (4CPU) : 3344 5 5874 57.1 % 3294 41/111
2.2s 3.9s
27 Gull 3 (4CPU) : 3341 5 5762 56.6 % 3295 40/111
2.1s 3.9s
28 Protector 1.9.0 (4CPU) : 3340 5 5734 56.6 % 3294 39/111
2.1s 3.9s
29 Stockfish 5 (1CPU) : 3335 5 5470 55.9 % 3294 41/111
2.4s 4.0s
30 Andscacs 0.87 (4CPU) : 3333 5 5598 55.4 % 3295 36/111
2.0s 3.9s
31 Houdini 4 (1CPU) : 3328 5 5181 54.6 % 3296 34/111
1.8s 4.0s
32 Fire 5 (1CPU) : 3327 5 5277 54.5 % 3296 33/111
1.7s 4.0s

33 DeepShredder 13 (1CPU) : 3326 5 5315 54.4 % 3296
36/111 2.2s 4.1s
34 Booot 6.1 (1CPU) : 3325 5 4521 54.1 % 3297 28/111
1.6s 2.7s
35 Komodo 8 (1CPU) : 3323 5 5271 53.9 % 3296 33/111
2.0s 4.0s
36 Andscacs 0.89 (1CPU) : 3318 5 5386 53.1 % 3296 34/111
2.3s 4.1s
37 Critter 1.6a (1CPU) : 3318 5 5234 52.9 % 3298 33/111
2.1s 4.1s
38 Andscacs 0.86 (4CPU) : 3317 5 5057 52.8 % 3298 32/111
2.1s 4.0s
39 Nirvanachess 2.3 (4CPU) : 3316 5 5031 52.8 % 3297 33/111
2.1s 4.1s
40 Equinox 3.30 (4CPU) : 3314 5 5062 52.1 % 3300 32/111
2.0s 4.0s
41 Equinox 3.30 (1CPU) : 3312 5 5062 51.7 % 3300 31/111
2.1s 4.1s
42 Naum 4.6 (4CPU) : 3308 5 4886 51.1 % 3300 33/111
2.6s 4.2s
43 Andscacs 0.87 (1CPU) : 3307 5 4894 51.1 % 3300 31/111
2.2s 4.1s
44 Fizbo 1.8 (1CPU) : 3305 5 5094 50.9 % 3299 28/111
1.8s 4.2s
45 Fritz 15 (1CPU) : 3303 5 4899 50.4 % 3300 30/111
2.2s 4.2s
46 Spark 1.0 (4CPU) : 3302 5 5287 50.2 % 3300 30/111
2.4s 4.3s
47 DeepRybka 4.1 (1CPU) : 3299 5 4834 49.5 % 3302 27/111
2.0s 4.2s
48 Andscacs 0.86 (1CPU) : 3298 5 4793 49.9 % 3299 29/111
2.2s 4.1s
49 Texel 1.06 (4CPU) : 3296 5 4927 49.4 % 3300 28/111
2.1s 4.3s
50 Hannibal 1.7 (1CPU) : 3295 5 4641 49.2 % 3300 24/111
1.2s 4.2s

51 Nirvanachess 2.3 (1CPU) : 3295 5 4815 49.2 % 3300 28/111
2.2s 4.3s
52 Arasan 19.2 (4CPU) : 3294 5 4961 48.8 % 3302 29/111
2.5s 4.3s
53 Gull 3 (1CPU) : 3293 5 4742 48.9 % 3300 26/111
1.8s 4.2s
54 Protector 1.8 (1CPU) : 3289 5 4564 48.3 % 3301 25/111
1.9s 4.0s
55 Protector 1.9 (1CPU) : 3289 5 4668 48.4 % 3300 25/111
1.8s 4.3s
56 Protector 1.6 (1CPU) : 3287 5 4611 47.8 % 3302 26/111
2.3s 4.2s
57 BlackMamba 2.0 (4CPU) : 3285 5 4507 47.7 % 3301 25/111
1.9s 4.2s
58 iCE 3.0 (1CPU) : 3283 5 4559 47.3 % 3302 24/111
1.9s 4.3s
59 Fizbo 1.7 (1CPU) : 3283 5 4729 47.3 % 3302 23/111
1.7s 4.3s
60 DeepGandalf 7.0 (2CPU) : 3282 6 4473 46.6 % 3305 22/111
1.6s 4.2s
61 Chiron 4 (4CPU) : 3278 5 4631 46.5 % 3303 24/111
2.3s 4.3s
62 RybkaWinFinder 2.2 (4CPU) : 3275 6 4525 45.7 % 3305
22/111 1.9s 4.3s
63 DeepJunior 13.3 (1CPU) : 3274 6 4482 45.6 % 3305 21/111
1.6s 4.3s
64 Chiron 3.01 (4CPU) : 3274 5 4528 45.6 % 3305 25/111
2.6s 4.4s
65 Nirvanachess 2.2 (1CPU) : 3274 5 4506 45.9 % 3302 23/111
2.0s 4.4s
66 Crafty 25.2 (4CPU) : 3273 6 4223 45.7 % 3303 19/111
2.2s 3.1s
67 Texel 1.06 (1CPU) : 3272 5 4566 45.5 % 3304 25/111
2.8s 4.5s
68 DeepFritz 10 (4CPU) : 3271 5 4617 45.2 % 3304 23/111
2.4s 4.4s

69 Chiron 2 (1CPU) : 3270 6 4550 45.0 % 3305 22/111
2.1s 4.3s
70 Wasp 2.00 (4CPU) : 3269 5 4507 45.0 % 3304 24/111
2.6s 4.5s
71 Rybka 3 (1CPU) : 3269 5 4350 45.0 % 3304 21/111
2.0s 4.3s
72 Naum 4.6 (1CPU) : 3269 6 4312 44.9 % 3304 20/111
1.6s 4.3s
73 Chiron 1.5 (1CPU) : 3267 6 4486 44.7 % 3304 22/111
2.3s 4.4s
74 Hakkapeliitta 3.0 (1CPU) : 3261 6 4453 43.8 % 3305 23/111
2.7s 4.4s
75 Texel 1.05 (1CPU) : 3260 6 4272 43.3 % 3307 22/111
2.4s 4.5s
76 Arasan 19.1 (4CPU) : 3260 5 4312 43.4 % 3306 21/111
2.4s 4.5s
77 LittleGoliath Revival (1CPU) : 3257 6 4219 42.7 % 3308 17/111
1.5s 4.1s
78 Chiron 4 (1CPU) : 3255 6 4404 42.8 % 3306 22/111
2.8s 4.5s
79 Arasan 19.2 (1CPU) : 3253 6 4254 42.5 % 3306 22/111
2.8s 4.5s
80 Vajolet 22.2 (1CPU) : 3253 6 4236 42.6 % 3305 22/111
2.8s 4.5s
81 Nimzo 8 (1CPU) : 3253 6 4252 42.1 % 3309 16/111
1.2s 4.3s
82 Arasan 19.1 (1CPU) : 3253 6 4315 42.4 % 3306 18/111
1.8s 4.4s
83 ZappaMexico II (1CPU) : 3249 6 3903 41.5 % 3309 15/111
1.1s 3.9s
84 TheKing 3.33 (1CPU) : 3247 6 4170 41.3 % 3309 16/111
1.6s 4.3s
85 Spark 1.0 (1CPU) : 3245 6 4276 41.4 % 3306 20/111
2.8s 4.6s
86 DeepFritz 10 (1CPU) : 3245 6 4106 40.9 % 3309 16/111
1.7s 4.4s

87 Naum 4 (1CPU) : 3245 6 3950 40.9 % 3308 15/111
1.2s 4.1s
88 TheKing 3.50 (4CPU) : 3243 6 4103 40.7 % 3308 18/111
2.3s 4.5s
89 Senpai 1.0 (1CPU) : 3241 6 4112 40.7 % 3307 17/111
2.0s 4.5s
90 Dragon 4.6 (1CPU) : 3241 6 3786 40.1 % 3311 14/111
1.6s 3.7s
91 Fruit 2.3 (1CPU) : 3238 6 3909 39.7 % 3311 15/111
1.4s 4.5s
92 Ruffian 1.0.1 (1CPU) : 3237 6 4116 39.8 % 3309 15/111
1.8s 4.5s
93 Spike 1.4 (1CPU) : 3228 6 4001 38.5 % 3309 15/111
2.3s 4.4s
94 RodentII 0.9.64 (1CPU) : 3227 6 3947 38.4 % 3310 17/111
2.8s 4.6s
95 Quazar 0.4 (1CPU) : 3226 6 3945 38.4 % 3309 15/111
2.2s 4.5s
96 Ufim 8.02 (1CPU) : 3226 6 3999 38.0 % 3311 14/111
1.9s 4.6s
97 Fruit 2.2.1 (1CPU) : 3224 6 3899 37.8 % 3311 13/111
1.5s 4.5s
98 Amoeba 1.2 (1CPU) : 3219 6 3810 37.0 % 3312 14/111
2.2s 4.5s
99 Junior 13.8Yokohama (1CPU) : 3219 6 3899 37.2 % 3310
12/111 1.1s 4.6s
100 Gandalf 6.0 (1CPU) : 3217 6 3785 36.8 % 3311 13/111
1.9s 4.4s
101 Anaconda 2.0.1 (1CPU) : 3217 6 3875 36.8 % 3311 16/111
2.8s 4.7s
102 Komodo 1.3 (1CPU) : 3216 6 3799 36.8 % 3310 14/111
2.1s 4.6s
103 Tao 5.7b (1CPU) : 3214 6 3778 36.5 % 3310 13/111
2.1s 4.3s
104 Tao 5.6 (1CPU) : 3211 6 3701 36.0 % 3311 12/111
1.7s 4.2s

105 Deuterium 14.3.34.130 (1CPU) : 3211 6 3742 36.0 % 3311

12/111 1.6s 4.5s

106 Wasp 2.00 (1CPU) : 3206 6 3767 35.3 % 3312 14/111

2.6s 4.7s

107 ProDeo 2.3 (1CPU) : 3201 6 3771 34.6 % 3312 10/111

1.3s 4.6s

108 Tao 5.4 (1CPU) : 3199 6 3629 34.2 % 3313 10/111

1.7s 4.1s

109 LoopMP 12.32 (2CPU) : 3198 6 3703 33.9 % 3314 10/111

1.6s 4.7s

110 Wasp 1.25 (1CPU) : 3197 6 3623 33.9 % 3313 13/111

2.6s 4.7s

111 Gaviota 1.0 (1CPU) : 3192 6 3641 33.4 % 3312 11/111

2.0s 4.7s

112 Laser 1.2 (1CPU) : 3191 6 3687 33.1 % 3313 10/111

1.9s 4.7s

113 ChessMind 0.82 (1CPU) : 3190 6 3613 33.0 % 3314 9/111

1.3s 4.5s

114 ProDeo 2.2 (1CPU) : 3188 6 3587 32.7 % 3313 11/111

2.8s 4.3s

115 SmarThink 1.70 (1CPU) : 3180 6 3495 31.4 % 3315 8/111

1.1s 4.6s

116 Ktulu 9 (1CPU) : 3180 6 3548 31.5 % 3314 8/111

1.4s 4.6s

117 Minko 1.3 (1CPU) : 3179 6 3503 31.4 % 3315 9/111

1.5s 4.7s

118 Murka 3 (1CPU) : 3175 6 3624 31.1 % 3313 8/111

1.9s 4.7s

119 Octochess r5190 : 3144 5 3387 27.1 % 3316 6/111

2.6s 4.8s

MST1 : Mean solution time (solved positions only)

MST2 : Mean solution time (solved and unsolved positions)

RIndex: Score according to solution time ranking for each position

AMD 8350-FX - 5sec/move - 512Mb Hash - Fritz 15 - Windows 10 / 64bit

Hier lässt sich der Output jeder Engine bei je-

der Aufgabenstellung downloaden (*Mai 2017 – PGN-Datei*).

Weiterführende Links

- Für jene Leser, die sich näher mit der Thematik Computerschach-Stellungstests befassen möchten, nachfolgend ein paar weiterführende Links:
- *Dr. Frank Schubert:* **Lösung eines alten Problems** – Autor *Schubert* untersucht zuerst die seinerzeit gängigen Auswerteverfahren verschiedener bekannter Stellungstests und stellt dann einen mathematisch neuen, dem FIDE-Elo-Verfahren ähnlichen Ansatz zur Differenzierung von Test-Ergebnissen vor. In einem historischen Exkurs wird auch Bezug genommen auf noch vor einigen Jahren gebräuchliche Tests von Autoren wie *Bednorz*, *Schumacher*, *Gurevich* oder *Scheidl*. Zum Schluss stellt Schubert seine eigene Methode vor, «welche die Schwächen der bisherigen Formeln beseitigt und erstmalig auf einer soliden schachlichen Theorie basiert.»
- *Lars Bremer:* **Was Stellungstests testen** – IT-Journalist und Programmierer *Bremer* repliziert darin auf den seinerzeit heftig umstrittenen CSS-WM-Test von *M. Gurevich*, wobei er ebenso unverhohlen wie amüsant in die Trick-Kiste greift, um seine Test-kritische These zu untermauern: Er löscht mit einem eigens dafür geschriebenen Tool in dem als *Gesamtheit*

konzipierten Test jeweils so lange einzelne Aufgaben, bis immer wieder Top-Resultate der zufällig gewählten (ggf. schwachen) Engine resultieren, womit *Bremer* den zufälligen Charakter von Test-Ergebnissen beweisen möchte. Aber da selbstverständlich ein umfangreicher und durchdachter Stellungstest immer *kompakt* gemeint ist, seine Aufgaben aufeinander abgestimmt sind und darum nicht einfach willkürlich zusammengestrichen werden können, ist der ganze Artikel eher satirisch denn wissenschaftlich einzustufen und darum ein einziger «Quatsch» (Zitat) – aber dennoch witzig zu lesen.

- *Lars Bremer*: **Chaos-System Deep Engine** – Ein sehr viel «seriöserer» und informativer Artikel des obigen Autors zur Problematik des «Zufälligen Zuges» bei Deep-Engines. Mit seinem «Fazit» bezüglich der Aussagekraft von Stellungstests bei MP-Rechnern ist der Schreibende zwar nicht einverstanden: Ausgedehnte Untersuchungen könnten sehr wohl dokumentieren, dass ein durchdachtes Design eines Stellungstestes diesen «MP-Effekt» zwar nicht restlos ausschalten, aber entscheidend abfedern kann, so dass er bezüglich Ranking schliesslich auch statistisch irrelevant wird. Doch Bremer erklärt das Phänomen aus der Sicht des Programmierers äusserst anschaulich und auch für Laien nachvollziehbar. Es wird erklärt, warum sich moderne «Deep»-Programme zuweilen völlig nicht-deterministisch, ja «chaotisch» verhalten beim

Ausspielen von Zügen.

- *Tord Romstad (Stockfish)*: Eine kurz zusammengefasste Erklärung dieses «MP-Effektes» findet sich auch in einem **Interview**, das *Frank Quisinsky* vor Jahren mit *Tord Romstad*, dem verantwortlichen Stockfish-Programmierer, sowie dessen Co-Autoren geführt hat. Zitat: «Wenn ein Schachprogramm eine Position, irgendwo tief innerhalb des Suchbaums, untersucht, macht es Gebrauch bzw. erinnert sich an frühere bereits untersuchte Positionen der gleichen Suche. Die Zugbeschneidung, Verkürzung oder Verlängerung hängen davon ab, welche Positionen vorher überprüft wurden und wie die Ergebnisse der Untersuchung dieser Positionen waren. Der Großteil der Informationen, der für eine Entscheidung verwendet wird, liegt im Arbeitsspeicher. Der Arbeitsspeicher steht allen Prozessoren zur Verfügung.
  Solange es nur einen Thread gibt, ist alles zu 100% reproduzierbar. Aber bei mehreren Threads beginnen seltsame Dinge zu geschehen, weil diese Threads nie synchron mit gleicher Geschwindigkeit aktiv sein können. Immer wieder wird eine CPU für ein paar Millisekunden eine Pause einlegen müssen und das Betriebssystem weist dann sofort eine andere Aufgabe zu. Das geschieht zufällig und ist nicht vorhersehbar, eine Kontrolle gibt es hierfür nicht. Als Konsequenz erreicht jeder Prozessor eine bestimmte Position eher zufällig und das wirkt sich dann auf die

Suche nach Entscheidungen zur aktuellen Position aus.»

- *Chess Programming Wiki:* **Test-Positions** – Diese informative Seite (anmeldepflichtig) verlinkt eine Vielzahl interessanter Artikel zur Thematik «Stellungstest» bzw. zu deren Autoren und stellt eine Fundgrube dar für jene, die sich mit der Materie näher befassen möchten. Gleichzeitig bietet die Webseite einen umfassenden Überblick auf die Historie und listet sogar interessante Forum-Beiträge aus früheren Jahren auf. ♦

Lesen Sie im Glarean Magazin auch über einen weiteren Stellungstest mit extrem schwierigen Schachaufgaben: Der Engine-Test «Nightmare 2»

Diesen Beitrag drucken oder downloaden oder mailen

Ähnliche Beiträge

Computerschach:

Computerschach:

Computerschach:

100 Endspiel-Puzzles für Programme
7. September 2007
In "Computerschach"

4. Version des Swiss-Test
8. August 2008
In "Computerschach"

Der Engine-Test «Nightmare 2»
25. August 2016
In "Computerschach"

Barbara Beuys: Maria Sibylla Merian (Biographie)

Das Sudoku-Quartett im März 2017

Kommentare sind willkommen! (E-Mail-Adresse wird nicht veröffentlicht)

Gib hier deinen Kommentar ein ...