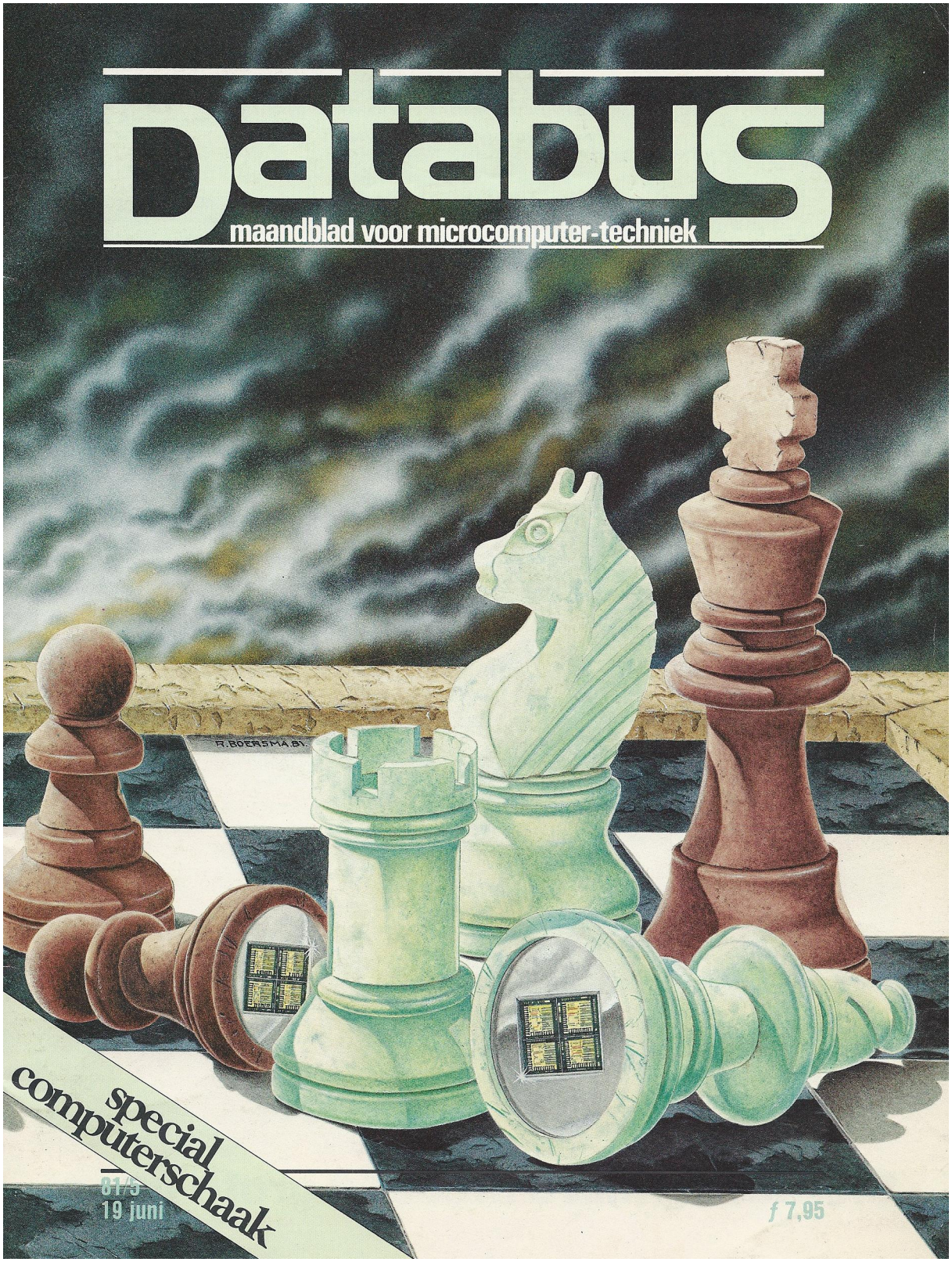

Databus (1981)

Wim Rens: Grondslagen van computerschaak

Grondslagen van computerschaak

Genereren, evalueren en selecteren

Wim Rens

Deze bijdrage beschrijft de grondslagen van nagenoeg alle schaakprogramma's die draaien op een microcomputer. Aansluitend wordt aandacht besteed aan de bij het programma Gambiet gevolgde strategieën. Dit is een schaakprogramma voor de TRS80 dat tijdens de vorig jaar gehouden kampioenschappen als derde eindigde.

Om te kunnen schaken moet een programma op een bord kunnen „kijken". De realisatie daarvan is eigenlijk nog het eenvoudigste van alles. Gambiet's schaakbord bestaat uit een serie computergeheugenlocaties, waarvan elke locatie overeenkomt met een veld van een echt bord. Op elke locatie kan een getal worden opgeborgen en verschillende getallen symboliseren verschillende stukken. Bijvoorbeeld: het getal 0 op een locatie betekent dat het bijbehorende veld leeg is; het getal 3 betekent dat er een witte loper op staat. Aldus is op elk moment de stelling nauwkeurig vastgelegd op 64 geheugenplaatsen.

Zet-generator

Van een schaakprogramma dat alleen maar op een bord kan kijken raakt niemand erg opgewonden. Men wil er ook nog eens een zet uitzien komen. Derhalve dient het programma in staat te zijn een lijst met alle mogelijke zetten te produceren. Een populaire methode is om die zetten te berekenen. Dit is niet zo'n vreselijk briljante gedachte, want computers zijn van huis uit nu éénmaal betere rekenwonders dan schaakwonders.

Het concept is simpel. Een voorbeeld: voorzie alle velden van het bord in de volgorde a1, b1, ..., h8 van de nummers 1 t/m 64 en iedere verplaatsing kan worden uitgeteld. Zo ligt het veld schuinrechts-voor altijd 9 velden verder. Ook Gambiet gebruikt deze techniek, waarvan de hoofdlijnen zijn:

- het bord wordt afgezocht naar hetzij witte, hetzij zwarte stukken;
- voor elk gevonden stuk wordt een bij dat stuk behorende tabel met bewegingsrichtingen geraadpleegd;
- alle mogelijke velden waarnaar het stuk zich in principe zou kunnen verplaatsen worden berekend;
- alle berekende verplaatsingen worden verder getest en bij gebleken legaliteit in een tabel met gegenereerde zetten geplaatst.

SOFTWARE KRONKELS

Zet-evaluator

Een programma dat legale zetten kan genereren kan al reglementair schaken. Naar willekeur wordt één van de zetten gekozen. Wij vrezen echter dat zo'n programma nogal eens zal verliezen. Het zal tenminste de goede van de slechte zetten moeten kunnen onderscheiden. Derhalve verzinnen we wederom iets met getallen: voor elke zet wordt een puntenwaardering afgeleid. Over de wijze waarop, de evaluatiefunctie, is veel discussie mogelijk. Gambiet's manier is ongeveer als volgt:

1 punt = zet leidt tot konings-winst
255 punten = zet leidt tot konings-verlies

De tussenliggende punten zijn voornamelijk gebaseerd op materiaalwinst of -verlies en in mindere mate op positionele voor- of nadelen.

Zet-selector

Een beginnend schaker wordt al direct geconfronteerd met „vooruitdenken". Ook een menselijke schaker heeft een soort evaluatiefunctie; althans hij gebruikt één of ander inzicht om een zet als goed of slecht te beoordelen. We weten allemaal dat zo'n inzicht nogal eens wordt bijgesteld als we de zet eventjes daadwerkelijk op het bord uitvoeren. Maar volgens de regels mag dat niet en we moeten ons beperken tot het uit het hoofd doorrekenen van varianten.
Voor een schaakprogramma is dat allemaal geen punt. Stukken worden over het geheugenbord verplaatst zoals het de computer uitkomt, of liever gezegd, zoals het de zet-selector uitkomt. Er is geen reglement dat dàt verbiedt.

Vooral de ontwikkeling van doelmatiger methoden om varianten door te rekenen heeft bijgedragen tot de vooruitgang van schaakprogramma's in het algemeen. De kracht van een computer is namelijk zijn tempo. Het programma „Belle" bekijkt 160.000 stellingen per seconde. Gambiet doet het met een paar minder: 75 stellingen per seconde...
Dus per zet van 3 minuten bekijkt Gambiet zo'n 13.500 stellingen. Dat zou prachtig kunnen zijn, ware het niet, dat misschien 13.400 van al die zetten door een normaal mens nooit in overweging zouden worden genomen. Hier hebben we dan meteen het zwakke punt van schaakprogramma's te pakken; hoog tempo prima, maar dan wel op een zinvolle wijze.
Hierna verklaren we twee principes, minimax en alfabeta, waarmee de zet-selector er toch nog het beste van tracht te maken.

Minimax

Stel de computer speelt met wit en kan drie zetten doen. We noteren die zetten als A, B en C.

Stel wit doet zet A, waarna zwart de keuze heeft uit drie tegenzetten, genoteerd A1, A2, A3.
Stel ook dat op zet B de antwoorden B1 t/m B3 mogelijk zijn en, u raadt het al, op zet C de antwoorden C1 t/m C3.
Op deze manier hebben we een soort boom van zetten geconstrueerd. Een boom met 3 takken en aan elke tak weer 3 uitlopers. Op elk van die 9 uitlopers laten we nu de evaluatiefunctie los.
Stel A1=50 punten, A2=100 en A3=20 punten. Welke zet zou zwart dus doen als wit zet A zou spelen? Juist, zet A2. Althans, als zwart verstandig speelt, maar met het uitgangspunt dat zwart erg dom speelt kom je waarschijnlijk niet zo ver.
Stel B1=80, B2=80, B3=90. Dus als wit zet B speelt dan speelt zwart B3 met 90 punten. Welke zet is nu beter voor wit, A of B? Zet B natuurlijk, want dan haalt zwart ten hoogste 90 punten i.p.v. 100 na zet A.
Voor wie dit allemaal niet direct gelooft, vermelden we dat echte wetenschappers dit hebben kunnen bevestigen en hiervoor de kreet *minimax* hebben bedacht. Wit wil het maximale en gunt derhalve zwart slechts het minimale. C'est la vie, n'est ce pas.
In de praktijk wordt natuurlijk een veel grotere boom geconstrueerd. En aan elke uitloper worden weer nieuwe uitlopers

Databus (1981) - Wim Rens: Grondslagen van computerschaak

geknoopt, en zo gaan we door tot we het welletjes vinden. Een gigantische taak, maar de computer is er goed voor.
Alle uiteindelijke uitlopers gaan we evalueren en dan klauteren we weer terug, al minimaximaliserend.

Moeilijkheidje
Aspirant schaakprogrammeurs die al direct met de minimax-methode aan de slag willen, zullen het volgende moeilijkheidje tegenkomen. Laten we er eens van uit gaan dat Gambiet twee zetten diep moet denken. Dat is helemaal geen zware eis, net genoeg bijvoorbeeld om mat in twee zetten te zien. Twee hele zetten (vier halve zetten) dat is een zettenboom met vier lagen (de vakterm voor „laag" is „ply"): 1e zet Gambiet, 1e zet tegenstander, 2e zet Gambiet, 2e zet tegenstander. Laten we er vervolgens eens van uit gaan dat er gemiddeld 32 zetten per beurt zijn te vinden. Dat betekent een zettenboom met 32 x 32 x 32 x 32 = ongeveer 1.000.000 uitlopers. Die moeten allemaal worden geëvalueerd. Gambiet doet er 75 per seconde; dat maakt 1000000/75 = 13333 s = 222 minuten. We praten hier dus over één zet. Voor een *„four ply depth search"* is dat wel wat lang.

Alfabeta
Duidelijk is dat we het snoeimes in de boom moeten gaan zetten; *„pruning"* heet dat in het Engels. Eén van de technieken daartoe is de α–β methode, die we weer illustreren aan de hand van onze zettenboom met de 9 uitlopers.

We weten al dat zet B betere resultaten oplevert dan zet A. Nu gaan we verder met de uitlopers van zet C. We evalueren eerst zet C1, en wat blijkt, zwart krijgt hiervoor 110 punten. Het is nu verder zinloos om de zetten C2 en C3 nog te evalueren. Wit zal namelijk zet C nooit doen, immers alleen al met zet C1 krijgt zwart meer punten dan op zet B mogelijk zou zijn. Dit is de gedachte achter „α–β pruning". Het aantal te onderzoeken varianten wordt aanzienlijk gereduceerd waarbij de computer tot dezelfde conclusie komt als wanneer hij alle varianten had doorgerekend.
Voorwaarde voor enige reductie is wel dat de goede zetten het eerst worden bekeken. Stel eens dat we eerst zet C2=40 punten, dan C3=40 punten, en dan pas zet C1=110 punten hadden bekeken. Dan hadden we nog bij de zetten C2 en C3 gedacht: dat ziet er gezond uit en rustig doorgeëvalueerd met variant C. Pas bij C1 hadden we de variant verworpen, maar ondertussen waren wel alle uitlopers bekeken. We staan met deze conclusie voor een aardig dilemma:

- We klauteren het snelst door de zettenboom als we eerst de beste en dan de slechtere zetten bekijken.
- We weten pas zeker welke de goede en welke de slechte zetten zijn als we zijn uitgeklauterd.

Door dit soort problemen wordt een serieus schaakprogrammeur op de rand van de waanzin gebracht. Maar om u gerust te stellen; er zijn vele van dit soort problemen.

Gambiet
In de schaakwereld wordt de term Gambiet gebruikt voor een openingszet, waarbij direct al een aantal pionnen worden geofferd teneinde een kansrijke aanval te kunnen opzetten. Het schaakprogramma Gambiet speelt niet alleen een sterk openingsspel, maar ook het verdere verloop van de partij valt meestal uit in het voordeel van de computer.
De aanzet tot het schrijven van een schaakprogramma was de aanschaf van een tot 100 stappen programmeerbare rekenmachine van Texas Instruments, die eigenlijk voor de aardigheid was gekocht. Later werd het de KIM en vier weken daarna de TRS80. Een soort logaritmische reeks dus, toenemend in hardwarekosten en afnemend in tussenpozen. Ondertussen had ik in het tijdschrift Personal Computing een aantal uitstekende artikelen over computerschaak gelezen. Ik vermoedde dat je daar iets langer mee zoet zou zijn dan met Mastermind, Nim en Bioritme. Dat vermoeden is waarheid geworden.

Wapenfeiten van Gambiet
De eerste speelklare versie van Gambiet heette nog Gabmol en was gereed omstreeks januari 1980. De ontwikkeling van het programma had ongeveer een jaar gekost. De hardware bestond uit een 16K TRS80 Model I, de systeemprogrammatuur uit een eenvoudige editor-assembler op cassette. Bijna twee C60-bandjes met source-tapes voor 9K byte object code betekende aanvankelijk zeer veel cassette-frustraties.
Maar goed, begin '80 was het dan zover. Het eerste wapenfeit vond plaats tegen IGM, het programma van Peter van Diepen. Peter reageerde op een oproep, geplaatst in de HCC Nieuwsbrief. Zijn programma deelt de landstitel met BS'66'77 van Barend Swets, dus Gabmol kon er meteen goed tegenaan. Nou dat deed het ook. Tot verbazing van zijn auteur werd IGM in 29 zetten verslagen. Gabmol bleek niet eens zwak.

De tweede mijlpaal op Gambiet's zegetocht werd bereikt onder toeziend oog van de firma Microtrend. Ten huize van één van hun directeuren stonden twee TRS80's tegenover elkaar. Eén geladen met Sargon II, de ander met Gabmol.
Ik wist dat Microtrend nog een schaakprogramma in hun collectie wilde opnemen en eind juni was het dan zover dat een bandje werd opgestuurd met het advies de wedstrijd maar eens te organiseren. Het resultaat mocht er zijn: op twee even trage Tandy computers had en heeft Gambiet geen moeite met die eens zo beroemde Sargon.
Tijdens de kampioenschappen in september deed Gambiet het weer niet slecht. Een derde plaats werd gedeeld met de kampioen van '79 (Sargon), die van '78 (Mike 3.0) en met Rook 4.0, een Zweeds programma op een razendsnelle Z8000. Nummer één werd een experimentele Chess Challenger, nummer twee een experimentele Boris.

Strategie
Een uitgangspunt bij het ontwerpen van Gambiet/80 was dat het zou moeten schaken als een computer. Dat wil zeggen, het zou louter gebruik maken van de *„brute force"*. De basis moest zijn een snelle zettengeneratie/evaluatie met behulp van uitgekiende Z80-machinetaal-instructies. Wanneer eenzelfde routine letterlijk tienduizenden keren wordt doorlopen, dan loont het de moeite er nog eens goed over na te denken of het niet een stapje korter kan. Om dit te illustreren onthullen we hier één van Gambiet's structurele kenmerken.
Het programma hanteert het schaakbordmodel, waarbij aan elk veld een geheugenlocatie wordt toegekend en waarbij voor de rand van het bord dummy-velden worden gecreëerd. Bij zo'n model zou een witte pion kunnen worden voorgesteld door +1, een zwarte door –1, een witte loper door +2, enz. Bij Gambiet is dit echter niet het geval. De bitpresentatie van de velden is als volgt:

	7654 3210
koning	x110 1111
dame	x000 1001
toren	x000 0101
loper	x000 0011
paard	x011 0011
pion	x001 0001
rand	1100 0000
leeg	0000 0000

Bit 7 geeft de kleur van een stuk aan: 0=wit, 1=zwart. Door deze presentatie is de volgende instructiereeks in principe mogelijk geworden:

```
LD    A, (VELD)
SLA   A
JP    Z, LEEG
JP    PO, RAND
JP    M, KONING
JP    C, ZWART
JP    WIT
```

Dus door één operatie, SLA, wordt voldoende informatie verkregen voor een selectie uit vijf alternatieven. Maar de presentatie bevat nog meer informatie. De

bits 3 t/m 0 geven steeds de materiële waarde. Voor een pion 1, een paard en een loper 3, een toren 5 en een dame 9. Bovendien is de overige structuur van het programma zodanig aangepast dat de bits 4 t/m 0 direct de ingang in een tabel met bewegingsrichtingen opleveren. Bewust is bit 5 alleen voor de koning en het paard gezet. Deze twee stukken onderscheiden zich in bewegingsgedrag van de overige stukken. De test of een veld bezet is geschiedt tenslotte via bit 0.

Het idee dat Gambiet moest schaken als een computer was ingegeven door het succes van de op *Shannon-A* strategie gebaseerde programma's als Sargon, Chess en Belle. Shannon-A wil zeggen, dat de variantenboom tot een beperkte diepte, maar over de gehele breedte wordt afgezocht. Gambiet rekent onder toernooi-condities (20…30 zetten/uur) tot en met de derde halve zet over de volle breedte van de boom, en over gereduceerde breedte tot en met de vijfde halve zet. Daarbij is de evaluatiefunctie vrij solide, althans materieel gezien. Het weldra vallen van een stuk wordt in de evaluatie meegenomen. Op een 1,77 MHz TRS80 is dat alles slechts mogelijk met een razendsnelle code en een minimum aan schaaktheorie. Van dubbelpionnen heeft Gambiet dan ook geen kaas gegeten. Gambiet kan redelijk combineren, maar enig theoretisch inzicht ontbreekt geheel.
Dat laatste bemerkt al spoedig de wat gevorderde clubspeler. Attent op de korte combinaties, maar met een goed plan op de wat langere termijn, zal hij gemakkelijk van Gambiet winnen. Zoniet andere computers. Zelf al over even weinig fantasie beschikkend, vinden ze in Gambiet een waardige tegenstander.

Verbeterde versie

Hoe nu voor Gambiet/80 een betere opvolger, Gambiet/81, te creëren? Het is mijn overtuiging dat je met het sneller maken van de software en de hardware wel wat, maar niet veel verder komt. Met veel moeite een zet dieper, maar dan nog zijn de gevolgen niet spectaculair. We praten hier natuurlijk wel over microcomputers en niet over Belle-achtige toestanden…
De werkelijke methode om een programma als Gambiet sterker te maken is er meer schaaktheorie – dus inzicht – in te brengen. Maar dat kost evaluatietijd, die moet worden beknot op het onderzoek naar de variantenboom. En dat mag weer beslist niet ten koste gaan van het vermogen tot combineren. Een mooie oplossing is de *Shannon-B* strategie die eenvoudig zegt: schaak ongeveer als een mens, onderken al die zinloze varianten en verwerp het onderzoek ernaar. Dit is natuurlijk wel mooi, maar niet eenvoudig. Met Shannon-B is het vallen en opstaan, dat hebben al veel schaakprogrammeurs moeten ondervinden. Eén van de betere prestaties van Gambiet/81 uit het recente verleden is hierbij afgedrukt (afb. 1). Het is een partij tegen mezelf. Bij het naspelen zal het de schaakkenner niet ontgaan dat het monster al duidelijk beter speelt dan zijn zwak begaafde schepper. Bovendien zal het de computerschaakkenner opvallen dat al die ellende wordt ingeluid met een schijnoffer op de 20e zet, terwijl computers in het algemeen – en Shannon-B in het bijzonder – toch moeilijk tot offeren zijn te brengen.

Dit artikel werd ook gepubliceerd in „Computerschaak", het orgaan van de Computer Schaak Vereniging Nederland. Door het storten van de contributie (*f* 40,- per jaar, KNSB leden *f* 30,-) op postgiro 4313210 t.n.v. CSVN Laren, kunt u lid worden van deze vereniging.

Afb. 1. Een demonstratiepartij tussen Gambiet/81 en de ontwerper ervan. Gambiet/81 speelt hier met zwart.

Wit	Zwart	klok Gambiet
1. d2-d4	Pg8-f6	00:00
2. c2-c4	g7-g6	00:00
3. Pb1-c3	Lf8-g7	00:00
4. e2-e4	d7-d6	00:00
5. Pg1-f3	o-o	00:00
6. Lc1-f4	Pb8-c6	01:23
7. h2-h3	Pf6-h5	03:54
8. Lf4-h2	Lc8-d7	04:20
9. g2-g4	Ph5-f6	05:45
10. Dd1-b3	Dd8-c8	15:57
11. o-o-o	Pc6-a5	20:28
12. Db3-c2	Pa5-c6	21:43
13. Lf1-g2	Pc6-b4	22:07
14. Dc2-b3	c7-c5	22:49
15. e4-e5	Lg7-h6+	24:33
16. Kc1-b1	Pf6-e8	26:07
17. Th1-e1	a7-a5	28:13
18. e5xd6	a5-a4	29:33
19. Db3-a3	Pe8xd6	31:22
20. Pc3-d5	Ld7-f5+	33:37
21. g4xf5	Dc8xf5+	34:04
22. Td1-d3	Pd6-c8	41:40
23. Pd5xe7+	Pc8xe7	42:34
24. d4xc5	Pb4xd3	43:25
25. Te1xe7	Pd3-b4+	47:13
26. Kb1-a1	Pb4-c2+	48:31
27. Ka1-b1	Pc2xa3+	48:33
28. Kb1-a1	Df5-b1±	48:35

Databus (1981) - Wim Rens: Grondslagen van computerschaak