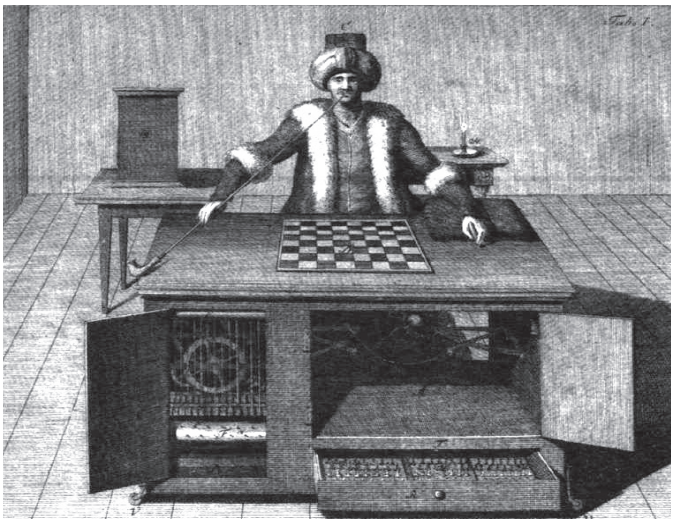
Computer schaak

Renze Steenhuisen, Collective Agent-Based Systems Group, Technische Universiteit Delft (j.r.steenhuisen@ewi.tudelft.nl)

Het schaakspel is altijd nauw betrokken geweest bij de ontwikkeling van de kunstmatige intelligentie. Toen kunstmatige intelligentie nog in de kinderschoenen stond, was het ontwikkelen van een computer die goed kon schaken één van de drijfveren achter het onderzoek.

De eerste schakende 'machine' dateert uit 1769, maar dit apparaat ('The Turk' geheten) bleek uiteindelijk een hoax te zijn. Halverwege de negentiende eeuw speculeerde de bekende Ada Byron (als programmeur van de Analytical Engine van Babbage) over de mogelijkheid dat machines zouden kunnen gaan schaken. Het was wachten tot - niemand minder dan - Claude Shannon een van de eerste artikelen publiceerde over hoe computers zouden kunnen schaken. In navolging hiervan ontwikkelde John von Neumann een programma dat kon schaken op een 6x6 bord, zonder lopers. In 1951 heeft uiteindelijk Alan Turing een programma ontwikkeld, al was het op papier, dat in staat was om een complete schaakpartij te spelen.

In dit artikel zullen we echter niet verder stil blijven staan bij de geschiedenis, maar we zullen ingaan op de technieken die zoal gebruikt worden in huidige computerschaakprogramma's. Het zal duidelijk worden dat een breed scala van technieken gebruikt kan worden uit verschillende deelgebieden van de informatica. Het voert te ver om hier al deze technieken en mogelijkheden te bespreken. Geïnteresseerde lezers worden aangemoedigd om verder te lezen en vooral zelf een schaakprogramma te ontwikkelen.



'The Turk' uit 1769

Basisbenodigdheden

Elk schaakprogramma heeft een basis nodig waarop de rest van het programma is gebouwd. Dit zijn de elementaire datastructuren en algoritmen die in verschillende vormen terug te vinden zijn in alle schaakprogramma's: een bordrepresentatie, zoekfunctie en evaluatiefunctie.

Omdat het bord de kern vormt waar het allemaal om draait, is het van groot belang dat hier een keuze wordt gemaakt op basis van snelheid. De mogelijkheden voor bordrepresentatie zijn legio, variërend van een eenvoudige 8x8 array tot meerdere bitboards of 16x8-representatie (de 0x88-representatie). Bitboards zijn 64-bit velden die per veld aangeven of een bepaalde eigenschap (zoals de aanwezigheid van een bepaald type stuk) waar is, of niet. Veel operaties kunnen daardoor met behulp van enkele logische operaties worden uitgevoerd. Bovendien geeft de stap van 32-bit naar 64-bit machines een directe snelheidswinst. Aan de andere kant maakt de 0x88-representatie het mogelijk dat de relatief dure controle op schaak zeer snel uitgevoerd kan worden.

Naast representatie is het duidelijk dat de bordrepresentatie gebruikt moet gaan worden om de zetten op te gaan doen. Door de verschillende legale zetten af te gaan en stuk voor stuk te doen, kunnen recursief alle mogelijke stellingen afgelopen worden. Deze wijze van het doorlopen van de toestandsruimte van schaken is een natuurlijke aanpak die ook door veel mensen toegepast wordt. Het enige wat nu nog ontbreekt is hoe we hieruit de beste zet selecteren voor een beginpositie. Dit kan eenvoudig gedaan worden door steeds de beste zet te kiezen voor de speler aan zet. Als winst voor zwart met een sterk negatieve waarde gepresenteerd wordt en winst voor wit met een sterk positieve, dan zal zwart de kleinste waarde prefereren en wit de grootste. Dit principe heet *mini-max*. Terwijl de beste zet gevonden wordt door alle volgende stellingen te bekijken, is het niet nodig om alle zetten af te lopen om deze beste zet te vinden. Het *alpha-beta* zoekalgoritme is een *branch-and-bound* algoritme dat hetzelfde resultaat geeft als het pure *mini-max* zoekalgoritme, maar dan heel veel sneller.

In principe hebben we nu genoeg, ware het niet dat een schaakstelling niet, zoals boter-kaas-en-eieren, helemaal kan worden doorgerekend. Er zijn naar schatting 10^{43} legale stellingen, wat te veel is om binnen een redelijke termijn te doorzoeken. Het is daarom nodig om ergens te stoppen met zoeken, zonder een eindsituatie bereikt te hebben. Omdat het voor deze stellingen niet mogelijk is om de theoretische (lees: exacte) waarde te bepalen, moeten ze een heuristische waarde toegekend krijgen. De waarde van deze afschatting wordt, net zoals bij Shannon in 1950, in moderne programma's gedomineerd door de materiaalbalans. Hierbij wordt de aanwezigheid van stukken gewaardeerd met waarden zoals deze te vinden zijn in leerboekjes voor beginnende schakers. Naast materiaal kunnen er natuurlijk ook veel andere belangrijke factoren uit de schaakliteratuur worden meegewogen.

Geavanceerde uitrusting

Nu we de basis voor een schaakprogramma hebben, zijn we toegekomen aan de delen van een schaakprogramma waardoor we daadwerkelijk de speelsterkte kunnen vergroten. Over het

algemeen wordt dit deel als het meest interessant en leukst gezien.

In het begin van elke schaakpartij staan de stukken in een vaste opstelling op het bord. Net als mensen kunnen ook computers in de openingsfase van het spel hun zetten proberen te baseren op kennis over openingen en ervaring uit het verleden. Dit is in vrijwel alle programma's geïmplementeerd als een grote database van stellingen met de bijbehorende beste zetten, maar dan handiger opgeslagen. Als een programma in een stelling komt die ook in zijn openingenboek staat, dan zal direct een zet gedaan kunnen worden uit het openingenboek. Dit bespaart kostbare rekentijd die in het middenspel goed gebruikt kan worden.

Al lijkt het misschien op het eerste gezicht eenvoudig om een dergelijk openingenboek samen te stellen, er doen zich enkele problemen voor die opgelost moeten worden. Enerzijds is het beter om een zo groot mogelijk openingenboek te hebben, anderzijds moet de inhoud van een hoge kwaliteit zijn. Eén van de problemen hierbij is dat programma's het niet altijd eens zijn met schakers van wereldklasse. Zo kan het voorkomen dat de laatste zet uit het openingenboek ongedaan wordt gemaakt in de volgende beurt, waarin het programma voor het eerst zelf moet beslissen. Dit is echter een zeer onwenselijke situatie, zoals je kan begrijpen. Een interessante mogelijkheid zou zijn om het programma zelf te laten bedenken wat goede openingen zijn.

Aan het eind van een schaakpartij kan het voorkomen dat er slechts enkele stukken op het bord staan. Ook hier kunnen computers, net als mensen, uitgerust worden met specifieke kennis en vaardigheden voor het optimaal uitspelen van bepaalde eindspelsituaties. Een voorbeeld hiervan is om gebruik te maken van de beschikbare eindspeldatabases, waarin optimaal spel is opgeslagen voor de situaties met maximaal zes stukken op het bord. Dit deel van het schaakspel is namelijk compleet uitgerekend/opgelost, en heeft ongeveer 1 terabyte opslagruimte nodig. Het moge duidelijk zijn dat niet iedereen zoveel ruimte beschikbaar heeft. Een compleet andere manier om deze eindspelkennis te gebruiken is het construeren van functies die hetzelfde bereiken. Het construeren van dergelijke functies blijkt echter een groot probleem te zijn.

Efficiënter zoeken

Met databases aan het begin en aan het eind zullen we ertussen toch moeten rekenen. In het voorgaande hebben we gezien dat zoeken een kwestie is van eenvoudig backtracken van (heuristische) evaluaties. Dit zoeken kan daarom op meerdere manieren verbeterd worden. Aan de ene kant hebben we de evaluatiefunctie die verbeterd kan worden, en aan de andere kant het zoeken zelf dat efficiënter gemaakt kan worden.

Om te beginnen kan de heuristische evaluatiefunctie verbeterd worden. We zullen nu drie verbeteringen bespreken. Ten eerste, omdat het materiaal dominant is in de evaluatie is het beter om veranderingen in materiaal, zoals slag- en promotiezetten, te zetten voordat de positie geëvalueerd wordt. Concreet betekent dit dat de statische evaluatiefunctie vervangen wordt door een dynamische, die eerst de slag- en promotiezetten doet voordat de statische evaluatiefunctie aangeroepen wordt. Dit zorgt voor een veel nauwkeurigere heuristische waarde. Ten tweede moet nauwkeurig nagegaan worden welke kenmerken van een stelling van belang zijn voor de statische evaluatie. Ondanks dat veel kenmerken in vrijwel elk programma voorkomen (zoals materiaal, plaatsingsbonus, koningsveiligheid en ontwikkeling), zijn er nog genoeg verschillen tussen de verschillende programma's. Ten derde moeten er uiteindelijk ook waarden toegekend worden aan de voorkomens van de gedetecteerde kenmerken. Veel mensen die (denken te) kunnen schaken zijn wellicht in staat schattingen te geven voor deze waarderingen. Mensen die wat minder goed zijn in het schatten van deze waarden, of daar gewoon geen zin in hebben, kunnen *machine learning* technieken toepassen om het programma automatisch waarden te laten vinden voor de verschillende kenmerken.

Naast de evaluatiefunctie kunnen we ook het zoeken zelf efficiënter laten verlopen. Ook dit kan weer op verschillende manieren bereikt worden. Eén van de belangrijkste verbeteringen is het zoveel mogelijk voorkomen van het veelvoudig doorzoeken van dezelfde posities. Als een positie bereikt wordt die al eerder tot een voldoende diepte doorzocht is, dan kan de toen berekende waarde gebruikt worden en hoeft er niet verder gezocht te worden. Het gebruik van deze transpositietabellen levert een behoorlijke besparing op doordat dezelfde posities bereikt kunnen worden door zetten in een andere volgorde na elkaar te spelen. Een tweede verbetering in de zoekfunctie is om 'interessante' stellingen beter en 'oninteressante' posities minder goed te bekijken. Dieper zoeken wordt bijvoorbeeld gedaan bij stellingen waarin een zet duidelijk beter is dan alle andere, of als er slechts één zet mogelijk is. Oninteressant zijn bijvoorbeeld die posities waarvan de waarde waarschijnlijk niet hoger wordt dan het beste resultaat, tot dan toe. Bij deze oninteressante posities kan dan besloten worden om ze helemaal niet verder te doorzoeken, of om ze minder diep te doorzoeken. Wees gewaarschuwd, want de speelsterkte van het programma kan snel afnemen als te snel besloten wordt om bepaalde stellingen niet verder te bekijken! Als laatste wijzen we erop dat de volgorde waarin de zetten bekeken worden van grote invloed is op de efficiëntie van het alpha-beta zoekalgoritme. Ter vergelijking, met de beste ordening kan een positie, in dezelfde tijd, twee maal zo diep doorgerekend worden dan met de slechtste ordening. Gegeven dat de speelsterkte toeneemt met zoekdiepte is het duidelijk dat een goede ordening van zetten veel bij kan dragen. Veel moderne programma's scoren 90% voor deze ordening.

(vervolg op pagina 27)

(advertentie)

ISB event 2006: AmIGro

16 maart 2006
Mediacentrale Groningen

ISB
INFORMATION SOCIETY & BUSINESS

AmIGro is een groot evenement voor KI-studenten, -onderzoekers en (ICT)bedrijven. Dit geeft je de kans om erachter te komen wat er op dit moment speelt op het gebied van KI en ICT, zowel in het onderzoek als in het bedrijfsleven.

Het evenement AmIGro bestaat uit drie onderdelen:

- **Ambient Intelligence Symposium**; hoofdsprekers Emile Aarts (Philips Research) en Kevin Warwick (Professor of Cybernetics, University of Reading, Engeland)
- **Exhibition Space**; bedrijven en kennisinstellingen presenteren hun laatste onderzoek
- **Carrière Markt**; spreek met bedrijven die geregeld vacatures hebben en oriënteer je op de arbeidsmarkt

Ambient Intelligence: Streven naar een intelligente omgeving. Apparaten passen zich aan aan de gebruiker en communiceren met elkaar. Bovendien verdwijnen de apparaten naar de achtergrond, de gebruiker staat centraal.

AmIGro brengt studenten, wetenschappers en bedrijfsleven samen. Bovendien is er speciale aandacht voor de KI aspecten van Ambient Intelligence zoals cognitieve ergonomie, multiagent systems en taal- en spraaktechnologie.

Zie voor meer informatie www.amigro.nl en het artikel over Ambient Intelligence in deze connectie. Ben je geïnteresseerd in het leveren van een bijdrage, bijvoorbeeld voor een workshop of het geven van een posterpresentatie over jouw onderzoek? Mail dan naar organisation@amigro.nl.

23 – 26 februari 2006
[Sonic Acts XI - The Anthology of Computer Art. Amsterdam]
In Paradiso / De Balie, Amsterdam vindt de elfde editie van het Sonic Acts festival plaats. Onderdeel ervan is een conferentie die een overzicht zal bieden van de computerkunst. Voor meer informatie:
http://www.sonicacts.com

25 - 26 april 2006
[Mind The Brain '06 UMC Utrecht]
Op dit symposium van de Utrechtse *Neuroscience & Cognition* Master zullen studenten van deze Master hun onderzoek presenteren. Tevens zullen verschillende gastsprekers uit dit veld vertellen over hun over hun onderzoek. Voor meer informatie:
http://www.mindthebrain.org

2 – 3 juni 2006
[RoboChallenge 2006 Groningen]
Voor meer informatie:
http://www.robochallenge.nl

juni 2006
[aimotion - BNAIS 2006 Nijmegen]
Voor meer informatie:
http://www.ru.nl/bnais

(vervolg van pagina 21)

Na ook deze technieken geïmplementeerd te hebben, kunnen we spreken van een behoorlijk schaakprogramma. We hoeven echter nog niet tevreden te zijn met de prestaties. Inmiddels moet duidelijk geworden zijn dat snelheid een belangrijk aspect is van een schaakprogramma. Hier wordt een mogelijkheid geboden om de specifieke computereigenschappen te gaan gebruiken. Een profiler kan je assisteren in het optimaliseren van die delen waaraan het programma relatief veel tijd kwijt is. Hier kan zelfs besloten worden om machine specifieke *assembler* instructies te gebruiken. Ook *multi-core* processoren en machines met meerdere processoren kunnen betere prestaties leveren door gebruik te maken van *multi-threading*. Ook kan je enthousiast geworden zijn om het programma geschikt te maken voor een supercomputer, of een cluster van computers. Met andere woorden, er zijn mogelijkheden genoeg.

Ondanks dat computerschaak al geruime tijd bestaat zijn er nog veel verbeteringen en onderzoek mogelijk. Veel technieken en ideeën uit computerschaak kunnen zonder grote problemen overgenomen worden naar andere vergelijkbare spellen, zoals dammen, go en vier-op-een-rij. Kortom, computerschaak is duidelijk een manier waarop een grote diversiteit van technieken uit de informaticaopleiding, vooral uit het AI-deel, spelenderwijs aangeleerd kan worden. De mogelijkheden van het toepassen van dergelijke technieken in computerschaak zijn zeker niet gelimiteerd tot de hierboven genoemde. Voor geïnteresseerden staan hieronder nog enkele referenties voor verdere verdieping. ⌀

Referenties:
- *http://en.wikipedia.org/wiki/Computer_chess*
- *http://www.st.ewi.tudelft.nl/~renze/chess.php*