Random search is no better in
general than systematic exhaustion,
and may introduce the possibility of failure.
It is tempting but irrational to look for
a panacea in chaos.

M. L. Minsky

(„Some Methods of Artificial Intelligence
and Heuristic Programming", Mechanisation of
Thought Processes, Vol. I, London 1959)

HEURISTISCHE PROGRAMMERING - EEN RECENTE ONTWIKKELING OP HET GEBIED VAN DE SIMULATIE[1])

door Drs. A. Bosman

1. *Inleiding*

Het begrip simuleren heeft twee betekenissen[2]). In de ene, ruime, betekenis houdt simuleren in: het weergeven (nabootsen, dupliceren) van de belangrijkste relaties die de inhoud en of het „gedrag" van een bepaald systeem beschrijven. Veelal zal deze weergave geschieden in de vorm van een mathematisch model. Noodzakelijk is dit echter niet, men denke in dit verband aan waterbouwkundige laboratoria en de trainingsapparatuur voor de opleiding van vliegers en chauffeurs. Elektronische rekenmachines, zowel digitale als analoge, zijn veel gebruikte hulpmiddelen bij de simulatie.

Simuleren wordt toegepast om één of meer van de volgende redenen.

1. Het systeem kan niet in een analytische vorm worden weergegeven, bepaalde onderdelen van het systeem wel.
2. Het systeem is wel in een analytische vorm weer te geven, maar in die vorm niet oplosbaar.
3. De vereiste informatie voor de oplossing van het systeem is niet, of niet in de juiste vorm, beschikbaar[3]).

Een belangrijke oorzaak van het optreden van één of meer van deze redenen is het feit dat de variabelen, die de relaties in het systeem bepalen, stochastisch van aard zijn. Met behulp van Monte-Carlo methoden tracht men door middel van het trekken van aselecte steekproeven de momenten van de verdelingsfuncties van deze stochastische grootheden te bepalen. In de andere, beperkte, betekenis wordt

---

[1]) De schrijver is dank verschuldigd aan Professor Dr. J. L. Meij, Drs. J. L. Bouma en Dr. H. G. Werkema voor hun waardevolle opmerkingen t.a.v. de inhoud van dit artikel.

[2]) Voor een methodologische beschouwing over het begrip simuleren, zie C. West Churchman: „An Analysis of the Concept of Simulation", *Symposium on Simulation Models: Methodology and Applications to the Behavioral Sciences*, eds. Austin C. Hoggatt en Frederick E. Balderston, Cincinnati 1963, pp. 1-3.

[3]) George W. Morgenthaler: „The Theory and Application of Simulation in Operations Research", *The Meaning, Scope and Methods of Operations Research*, ed. Russell L. Ackoff. New York 1961, pp. 372-376, geeft nog een aantal andere argumenten voor de toepassing van simulatietechnieken. Deze argumenten kunnen alle worden herleid tot de hier genoemde drie.

simuleren dan ook wel vereenzelvigd met het toepassen van deze Monte-Carlo methoden[4]).

Kenmerkend voor de toepassing van simulatietechnieken is dat een *beschrijving* van een systeem, of een onderdeel daarvan, wordt verkregen. Een dergelijke beschrijving vormt de basis voor het vinden van een oplossing voor problemen die zich in het betreffende systeem kunnen voordoen. Een oplossing kan intuïtief worden gevonden of door de toepassing van bepaalde technieken, bv. die van het operationele onderzoek, worden bepaald. Alhoewel de gegevens die uit de simulatie resulteren voor het vinden van deze oplossingen dikwijls onmisbaar zijn, worden de simulatietechnieken doorgaans niet gebruikt om bij het zoeken hiernaar te assisteren. Oplossingen worden gevonden door de toepassing van andere technieken op het uit de simulatie verkregen materiaal.

Bij de hier te bespreken recente ontwikkelingen op het gebied van de simulatie, met name die van de heuristische programmering, tracht men juist met behulp van de simulatie oplossingen voor bepaalde problemen te vinden. Het betreft hier vooral problemen die wel in een analytische vorm kunnen worden weergegeven, maar in die vorm niet kunnen worden opgelost. Het aantal toelaatbare oplossingen is nl. zo groot, dat het onmogelijk of ondoenlijk is te trachten hieruit de optimale te berekenen. De klemtoon valt dus niet op de beschrijving, maar op het *zoeken* van een „goede" oplossing uit een groot aantal mogelijke.

De heuristische methoden en de Monte-Carlo methoden maken beide deel uit van het arsenaal van simulatietechnieken. Beide voorzien in een geheel verschillende behoefte. De Monte-Carlo methoden in die van het leveren van een adequate beschrijving van een systeem, de heuristische methoden in die van het beperken van de zoekactiviteiten bij het vinden van een oplossing.

In de volgende paragraaf zal een nadere omschrijving worden gegeven van de problemen waarop de heuristische programmering kan worden toegepast. De gedachte dat dit alleen zinvol mogelijk is indien geen optimale oplossingen kunnen

---

[4]) Het is in het kader van dit artikel niet mogelijk alle gebruikte begrippen te definieren of van een toelichting te voorzien. Enkele begrippen zullen, omdat zij wellicht voor vele lezers niet zonder meer duidelijk zijn, worden omschreven.

Een systeem is in een analytische vorm weergegeven indien de onderdelen ervan in bepaalde relaties kwantitatief kunnen worden beschreven, waarbij uit deze relaties tevens het verband tussen de onderdelen moet blijken. Het is in het algemeen alleen de mathematische formulering die aan deze twee voorwaarden voldoet. Men kan twee soorten systemen onderscheiden.

Het gedetermineerde systeem, waarin de relevante grootheden op meetfouten na zijn bepaald en het stochastische systeem, waarin de waarde van deze grootheden afhankelijk is van een kansverdeling. Deze kansverdeling wordt weergegeven door een zgn. verdelingsfunctie, die de grootte van de kans aangeeft dat de stochastische grootheid hoogstens gelijk is aan een gegeven reëel getal. De momenten van een kansverdeling kunnen worden gebruikt voor een globale beschrijving van de vorm en de ligging ervan. Het eerste moment is de zgn. mathematische verwachting - het gemiddelde - en het tweede moment is de variantie. Deze laatste geeft een maatstaf voor de spreiding.

Bij de toepassing van de Monte-Carlo methoden gaat men veelal uit van bepaalde kansverdelingen van waargenomen grootheden, waarbij men de verdelingsfunctie van de stochastische variabelen uit het systeem aanpast. Door het gelijktijdig optreden van meerdere grootheden met een kansverdeling kan dikwijls een oplossing langs analytische weg niet worden gevonden. Men gaat dan met behulp van het systeem de werkelijkheid nabootsen - simuleren - om te trachten een oplossing te vinden. Bij deze nabootsing doen zich een aantal problemen voor waarvan er hier twee belangrijke worden genoemd. Nabootsen, betekent het uitvoeren van een steekproefexperiment, de resultaten hiervan zijn stochastisch en bezitten derhalve een variantie. Eén van de moeilijkste problemen van de Monte-Carlo methoden is het nabootsingsproces zo uit te voeren dat deze variantie binnen redelijke grenzen blijft. Het andere probleem is het produceren van een verdelingsfunctie die aansluit bij de kansverdeling van de waargenomen grootheden.

worden gedefinieerd is onjuist. Het onderscheid tussen problemen met wel of geen optimale oplossingen is slechts gradueel, en gezien vanuit een economisch gezichtspunt zelfs niet relevant. In de daaropvolgende paragraaf zal een korte beschrijving van de heuristische programmering worden gegeven, terwijl in de laatste paragraaf deze programmering aan de hand van een voorbeeld nog nader zal worden toegelicht.

2. *Het relatieve van het optimale*

In de economie denkt men veelal in termen van maxima en minima, meer in het algemeen in optimale toestanden van bepaalde systemen. Dit geldt ook voor de bedrijfseconomische theorie. Het in deze theorie gebezigde kostprijsbegrip kan bv. niet worden gedefinieerd als men niet een aantal optima, zoals de meest geëigende produktiemethode, noodzakelijk voor de berekening van deze kostprijs, zou kunnen bepalen. De vraag rijst echter of men hierbij niet uitgaat van een stilzwijgende veronderstelling, waaraan dikwijls niet kan worden voldaan, nl.dat de noodzakelijke optima ook inderdaad kunnen worden berekend. Te weinig heeft men zich in de economische theorie gerealiseerd dat het dikwijls onmogelijk of ondoenlijk is om in een bepaald systeem een optimum te vinden. Hier wreekt zich de hoge graad van abstractie van bepaalde uitgangspunten in deze theorie, indien men althans toepassing ervan beoogt.

Een van deze punten is de gebruikelijke veronderstelling t.a.v. de vorm van de produktiefunctie. Door de keuze van deze vorm wordt een aantal problemen, bv. op het gebied van de produktieplanning, niet onderkend. Dit is ondermeer het geval bij de zg. problemen van combinatorische aard.

Dergelijke problemen treden op wanneer niet eenduidig het verband kan worden bepaald tussen een aantal samenhangende grootheden[5]). Een voorbeeld hiervan is, hoe moeten vijf orders die alle op vijf verschillende machines bewerkingen moeten ondergaan, waarbij de volgorde van bewerking geen rol speelt, over deze vijf apparaten worden verdeeld zodanig dat de totale tijd, noodzakelijk voor de produktie, minimaal is. Er bestaan 25 miljard mogelijke combinaties[6]). Zou men elke combinatie in één seconde kunnen berekenen, wat met een elektronische rekenmachine een eenvoudige opgave is, dan zou de machine, indien 24 uur per dag werd gedraaid, bijna 8 eeuwen bezig zijn voordat de optimale oplossing zou zijn gevonden[7]). Het blijkt met behulp van de meest moderne hulpmiddelen niet mogelijk voor zelfs zeer eenvoudige problemen van combinatorische aard optimale oplossingen aan te geven. Toch gaat men in de algemene economie en de bedrijfseconomie - met name in de leer van de kwantitatieve verhoudingen - uit van

[5]) Combinatorische wiskunde is die tak van de wiskunde die zich bezig houdt met de verdeling van elementen over verzamelingen. Herbert John Ryser: *Combinatorial Mathematics*, The Carus Mathematical Monographs nr. 14, t.a.p., p. 2.

[6]) In het geval dat men één order over vijf machines moet verdelen, waarop deze orders achtereenvolgens een bewerking ondergaan, ontstaan er 5! permutaties of mogelijkheden. Voegt men een tweede aan de eerste order toe dan ontstaan er, ervan uitgaande dat men de eerste order eerst aan een machine toewijst, nogmaals 4! permutaties of mogelijkheden. Op deze wijze door redenerende zou men voor 5 orders $5! \times 4! \times 3! \times 2!$ mogelijke combinaties krijgen. Dit aantal is aanzienlijk kleiner dan de 25 miljard door Pounds genoemd. De fout die men in dat geval echter maakt is, dat men één order als uitgangspunt van de mogelijke verdeling kiest en daar dan steeds een nieuwe order aan toevoegt. Men kan echter elke order als startpunt voor de verdeling nemen. Dan ontstaan er $5! \times 5! \times 5! \times 5! \times 5!$ mogelijkheden.

[7]) William F. Pounds: „The Scheduling Environment", *Industrial Scheduling*, eds. John F. Muth en Gerard L. Thompson, Englewood Cliffs 1963, p. 6.

produktiefunkties, waarvan wordt verondersteld dat de daarin voorkomende combinatorische problemen kunnen worden opgelost. Ook in de leer van de interne organisatie besteedt de bedrijfseconomische theorie geen aandacht aan de oplossing van dit soort problemen[8]).

Naast deze combinatorische problemen staan andere, waarvoor wel optimale oplossingen kunnen worden aangegeven. Men beschikt dan over technieken, bv. die van de lineaire programmering, waarmee onder bepaalde omstandigheden binnen een redelijke tijd uit een aantal mogelijke oplossingen de optimale kan worden gekozen[9]). De op deze wijze gevonden optima zijn echter relatief, zij gelden slechts in het kader van het systeem dat werd gebruikt.

Een systeem geeft per definitie niet de gehele werkelijkheid weer. Dit betekent dat er een mate van gespecificeerdheid van het systeem bestaat die, rekening houdende met de kosten en de opbrengsten bij toepassing ervan, optimaal is. Het bepalen van dit optimum betekent doorgaans het oplossen van een combinatorisch probleem. Er zijn geen aanwijzingen die de veronderstelling rechtvaardigen dat dit laatste combinatorische probleem gemakkelijker zou kunnen worden opgelost dan de hiervoor genoemde.

Er bestaan dan ook tussen de problemen met een zg. optimale oplossing en die waarvoor het niet mogelijk is een dergelijke oplossing binnen een redelijke tijd te vinden geen principiële verschillen. Het graduele verschil wordt bepaald door het antwoord op de vraag in hoeverre de schijnbaar optimale oplossingen afwijken van de met behulp van de simulatie gevonden niet optimale oplossingen. Het antwoord op deze vraag zal worden bepaald door het feit of men er in slaagt efficiënte technieken voor het toepassen van de heuristische methoden te ontwikkelen. De heuristische programmering biedt wat dit punt betreft de meeste perspectieven[10]). De op economisch gebied met deze programmering bereikte resultaten

---

[8]) Pounds, t.a.p. p. 7, merkt op dat hij tijdens een onderzoek bij een aantal bedrijven in de V.S. tot de ontdekking is gekomen dat het „scheduling" probleem in de praktijk niet als een combinatorisch probleem wordt gezien. Dit zou een verklaring kunnen zijn voor het negeren van dit probleem in de economische theorie. Daar staat echter weer tegenover dat de praktijk, dit in tegenstelling tot de theorie, de economische problemen veelal niet bewust als optimalisatieproblemen opvat.

[9]) Een lineair programmeringsprogramma beschrijft een systeem waarin lineaire relaties, in de vorm van gelijkheden en ongelijkheden, tussen de variabelen bestaan. Voorts kent men een lineaire functie, de zgn. waardefunctie, waarin de betekenis van elk der variabelen voor het systeem wordt aangegeven. Deze functie moet worden geoptimaliseerd (maximum of minimum) onder de voorwaarden dat de variabelen niet negatief zullen worden en dat aan de gegeven relaties wordt voldaan. Het voor de economist relevante aspect van de lineaire programmering is dat expliciet rekening wordt gehouden met het element van de schaarste. Immers een ongelijkheid geeft weer dat van een bepaald goed (bv. grondstof of machinecapaciteit) niet meer, wel minder, kan worden gebruikt dan de maximum aanwezige hoeveelheid. Het is juist dit schaarste-element en de verschillen in relatieve importantie der variabelen, die het keuzenprobleem complex maken. Er bestaat een uitgebreide literatuur over de lineaire programmering. Voor de Nederlandse literatuur, zie J. L. Meij: *Theoretische bedrijfseconomie*, dl. I, Den Haag 1960, pp. 272-284; I. van der Zijpp: *Lineaire Programmering in het Bedrijf*, Leiden 1964.

[10]) De heuristische programmering is slechts één van de middelen, zij het een zeer efficiënte, om de heuristische methoden toe te passen. Zie, voor een algemene beschouwing over de heuristische methodiek, G. Polya: *How to Solve it*, New York 1957.

doen vermoeden dat ook het graduele onderscheid tussen beide soorten van oplossingen steeds meer zal vervagen[11]).

3. *De heuristische programmering*

Heuristische programma's geven geen optimale oplossingen, hierin ligt het verschil met de algoritmische programma's en de overeenkomst met andere simulatietechnieken[12]). Het heuristische programma verschilt echter van deze laatste omdat het geen beschrijving geeft van een systeem, maar van de wijze waarop men een probleem, dat zich daarin zou kunnen voordoen, kan oplossen. Simplistisch gesteld zou men kunnen zeggen dat men bij het opstellen van een heuristisch programma tracht de wijze waarop een mens een bepaald probleem oplost in de vorm van instructies voor machines te vertalen[13]). Terecht spreekt De Groot dan ook van geprogrammeerde denkmethoden[14]). De machineinstructies (het programma) bevatten „overwegingen" die een mens bij het oplossen van een bepaald probleem in ogenschouw neemt, alsmede een bepaalde kwantificering van de relaties tussen deze „overwegingen". Men simuleert wanneer men tracht, op de hierna te behandelen wijzen, een oplossing voor het gestelde probleem te vinden. Wat men simuleert is dus een wijze van oplossen van problemen[15]).

De problemen die zich voordoen bij de opstelling van een heuristisch programma zijn, wegens het ontbreken van een achterliggende theorie, voornamelijk van programma-technische aard. Aan bepaalde voorwaarden moet echter worden voldaan,

11) De heuristische programmering staat op dit moment nog in „de kinderschoenen". Het aantal toepassingen is gering. Dit neemt niet weg dat de weg die men met deze vorm van programmering is ingeslagen een veel belovende kan zijn. Immers steeds meer blijkt dat de combinatorische problemen die zich bij de planning voordoen omvangrijk zijn en dat de gebruikelijke methodieken hiervoor geen oplossing bieden. Veel research op het gebied van de produktieplanning, met name die waarbij men gebruik maakt van netwerktechnieken, tendeert in de richting van heuristische programmering, alhoewel doorgaans andere omschrijvingen worden gebruikt. Zie, Robert J. Blair: „Critical Path Resources Simulation and Scheduling", *IEEE Transactions on Engineering Management,* september 1963; B. L. Fry: „Scans-System Description and Comparison with PERT", *IRE Transactions on Engineering Management,* september 1962. Ook het werk van bijvoorbeeld Wiest en Thompson, in het kader van de netwerkplanning, gaat in de richting van de heuristische programmering. Zie, B. Giffler and G. L. Thompson: „Algorithms for Solving Production-Scheduling Problems", *Operations Research,* Vol. 8, nr. 4 en Jerome D. Wiest: „Some Properties of Schedules for Large Projects with Limited Resources", *Operations Research,* Vol. 12, nr. 3., pp. 395-419. Zie voorts de artikelen van Thompson e.a. in één van de voetnoten aan het einde van deze paragraaf. Het boek *Industrial Scheduling,* t.a.p., bevat eveneens een aantal artikelen op dit gebied. Ongetwijfeld de bekendste studies op het terrein van de heuristische programmering, liggende in het economische vlak, zijn die van, Fred M. Tonge: „Summary of a Heuristic Line Balancing Procedure", *Computer and Thought,* ed. E. A. Feigenbaum en J. Feldman, New York 1963, pp. 168-191 en G. P. E. Clarkson: *A Simulation of Trust Investment,* Englewood Cliffs 1961.

12) Een algoritmisch programma geeft eenduidig bepaalde oplossingen. Dit is bv. het geval met de programma's die in de administratieve sfeer worden gebruikt en met die, welke in het kader van het operationele onderzoek doorgaans worden toegepast. Generaliserend zou men kunnen stellen dat het merendeel van de operationele onderzoek technieken juist worden ontwikkeld om eenduidig bepaalde - optimale - oplossingen te kunnen aangeven. Voor een uitvoeriger uiteenzetting over het verschil tussen heuristische en algoritmische programmering wordt men verwezen naar A. Bosman: „De Elektronische Rekenmachine: Rekenen en Redeneren", *Bedrijfseconomische Verkenningen,* red. J. L. Bouma en H. Willems, Den Haag 1965, pp. 150-170.

13) Uiteraard maakt men daarbij gebruik van de bijzondere eigenschappen van de elektronische rekenmachine.

14) A. D. de Groot: Methodologie, Den Haag 1961, p. 357.

15) Simon en Newell achten dit aspect van zodanige importantie, dat zij voorstellen om i.p.v. simuleren het woord realiseren te gebruiken. Bij simuleren denkt men aan het nabootsen van fysieke systemen, wat bij de heuristische programmering niet het geval is. Allen Newell and Herbert A. Simon: „The Logic Theory Machine, a Complex Information Processing System", *IRE Transactions on Information Theory,* sept. 1956.

alvorens men een dergelijk programma kan opstellen. Deze voorwaarden zijn de volgende.

a. Het doel dat men met het programma tracht te bereiken, de oplossing van een bepaald probleem, moet zo kunnen worden omschreven dat mogelijke oplossingen van het probleem kunnen worden onderkend.
b. Men moet een zodanig inzicht in het probleem hebben dat kan worden aangegeven welke gegevens noodzakelijk zijn om een oplossing te kunnen vinden. Hierdoor wordt de invoer van de data bepaald. Wordt aan beide voorwaarden voldaan dan impliceert dit dat het systeem, waarbinnen het probleem dat men wenst op te lossen zich voordoet, bekend is.

De heuristische programmering steunt op drie grondprincipes.

A. Het principe van de *probleemtransformatie.* Bij de oplossing van een probleem staan tegenover elkaar aan de ene kant de ingevoerde data, aan de andere kant één of meer gespecificeerde doelstellingen. Het probleem luidt, transformeer deze data in een zodanige vorm dat aan de doelstelling wordt voldaan. De gebruikelijke methodiek hierbij is dat men tracht de doelstelling te ontleden in een aantal sub-doelstellingen. Hierbij gaat men er vanuit dat de ingevoerde data gemakkelijker in deze sub-doelstellingen kunnen worden omgezet dan in de doelstelling zelf. Uiteraard geldt hierbij de eis dat de som van de sub-doelstellingen de eigenlijke doelstelling moet opleveren.

B. De *transformatie van de doelstelling.* Dit kan op twee verschillende manieren geschieden.

1. Kan men een probleem oplossen met behulp van een vaststaand aantal regels waartussen het verband bekend is, zoals bv. bij problemen uit de propositielogica of bij die van het schaakspel, dan zal men met behulp van deze regels de sub-doeleinden door het programma zelf laten bepalen. Het probleem bij de opstelling van het programma is dan, welke regels moeten worden opgenomen en welke heuristische principes (zie punt C) moeten worden toegepast om zo efficiënt mogelijke sub-doeleinden te krijgen.

2. Is het aantal regels niet bepaald, het verband er tussen niet of niet exact bekend of heeft men slechts de beschikking over bepaalde vuistregels, dan zal men bij de opstelling van het programma de sub-doeleinden vooraf bepalen. Het probleem is dan zo efficiënt mogelijke sub-doeleinden te vinden, rekening houdende met de bekende regels en de heuristische principes.

   In beide gevallen is er dus sprake van het toepassen van bepaalde regels, hier verder *operaties* genoemd, voor het vinden van een oplossing. In het eerste geval *ontstaan* door toepassing van deze operaties sub-doeleinden, in het tweede geval worden de ingevoerde data via de operaties omgezet in *gegeven* sub-doeleinden. De vraag welke operaties in een bepaald geval moeten worden toegepast is afhankelijk van de mogelijkheden die het programma biedt om het verschil tussen de ingevoerde data aan de ene kant

en de sub-doeleinden c.q. doeleinden aan de andere kant te kunnen onderkennen (zie de volgende paragraaf)[16]).

C. Het derde grondprincipe betreft de zgn. *heuristische principes*. Deze zijn noodzakelijk om een aantal problemen van programma-technische aard te kunnen oplossen. Twee van de bekendste hiervan zijn:

1. Indien er op een bepaald moment meerdere operaties ter beschikking staan, moet er een richtlijn zijn op grond waarvan een keuze uit deze operaties kan worden gedaan[17]).
2. Indien een bepaalde operatie wordt toegepast, kan deze toepassing worden beëindigd indien: het gezochte resultaat wordt gevonden, er geen verdere toepassingsmogelijkheden meer zijn of er een bepaalde richtlijn wordt opgenomen die aangeeft hoe vaak een bepaalde operatie moet worden toegepast, bv. het aantal zetten dat tijdens een schaakspel wordt beschouwd, alvorens het programma naar een andere operatie of een ander probleem overstapt. Die andere operatie of het andere probleem moeten door het programma worden aangegeven.

Heuristische programma's kunnen voor vele doeleinden worden gebruikt. Men zou hier een onderscheid kunnen maken tussen analytische en synthetische toepassingsmogelijkheden. Bij de analytische toepassingsmogelijkheden maakt men gebruik van heuristische programma's om met behulp van inductie een bepaald „gedrag" van een systeem te verklaren. Uitgaande van bepaalde relaties lost men problemen die zich binnen deze relaties kunnen voordoen op en legt de resultaten daarvan, het „gedrag", vast. Men simuleert als het ware het „gedrag" van een systeem uitgaande van bepaalde relaties. In deze vorm wordt de heuristische programmering gebruikt bij de opstelling van een verklarende theorie voor het „gedrag" van de bedrijfshuishouding, o.a. door Cyert en March[18]). Bij de synthetische mogelijkheid tracht men met behulp van deductie de onderdelen van een systeem zodanig te laten samenwerken dat dit beter functioneert. Essentieel hierbij is dat het heuristische programma optreedt als een soort *zoekprogramma* naar mogelijk goede of betere oplossingen. Men zoekt deze oplossingen door middel

---

16) De mogelijkheden voor de praktische toepassing van de heuristische programmering worden bepaald door de aard van de oplossing die men zoekt. Men kan hier twee situaties onderscheiden. In het eerste geval is de oplossing eenduidig bepaald, bv. in de vorm van één getal of van een verzameling van getallen. Dit is het geval in het in de volgende paragraaf te behandelen voorbeeld, zij het dat het daar geen getallen maar symbolen betreft. In het tweede geval wordt de oplossing zelf niet aangegeven, maar wel de voorwaarden waaraan deze moet voldoen. De kans een eenduidig bepaalde oplossing te vinden is in het algemeen kleiner dan de kans een oplossing te vinden die aan bepaalde voorwaarden voldoet. Dit betekent dat het programma voor het zoeken naar een oplossing van de eerste soort doorgaans aan hogere eisen, voor wat betreft de drie grondprincipes en dit met name voor de laatste twee, moet voldoen dan een programma voor het vinden van een oplossing van de tweede soort. Hierin ligt dan ook de ratio voor het opstellen van dit soort programma's; de grondslagen van de heuristische programmering worden met behulp ervan ontwikkeld. Zou dit niet het geval zijn dan zou het opstellen van een programma voor het zoeken van een oplossing die men reeds kent weinig zin hebben.

17) Hierbij kan men gebruik maken van het zgn. leerproces, wat in dit geval zou kunnen betekenen dat men de rekenmachine laat registreren wanneer een bepaalde operatie met succes is toegepast. De operatie met de hoogste frequentie zal dan in overeenkomstige situaties het eerst worden gekozen.

18) Richard M. Cyert and James G. March: *A Behavioral Theory of the Firm*, Englewood Cliffs 1963.

van simulatie[19]). Het „gedrag" van het systeem wordt gesimuleerd uitgaande van bepaalde beleidsregels. Deze regels kunnen, noodzakelijk is dit echter niet (zie noot 19), geheel of gedeeltelijk overeenstemmen met de relaties genoemd bij de analytische toepassingsmogelijkheden. Het zijn deze regels en de wijze waarop zij elkaar onderling beïnvloeden die de efficiency van het zoeken bepalen.

Een opmerking van Minsky, als opschrift van dit artikel gebruikt, wordt de laatste tijd door de feiten bevestigd[20]). Het blijkt dat de oplossing van het zoekprobleem eerder moet worden gezocht in de richting van de heuristische programmering dan in het toepassen van de Monte-Carlo methoden.

4. *Een voorbeeld*[21])

Het voorbeeld van simulatie dat hier wordt behandeld, is ontleend aan een toepassing van het door Newell, Shaw en Simon opgestelde programma, de zgn. General Problem Solver (GPS). Dit is één van de bekendste heuristische programma's, waarbij bij de transformatie van de doelstelling gebruik wordt gemaakt van een aantal vaststaande regels (sub B — punt 1, uit de vorige paragraaf)[22]). Het voorbeeld heeft betrekking op een probleem uit de propositielogica[23]). Interessant in dit geval is dat met elkaar wordt vergeleken de resultaten van een hardop denkende proefpersoon en die van de rekenmachine. Uit een vergelijking van de resultaten blijkt duidelijk de overeenkomst in de pogingen van beide om een oplossing te vinden.

---

[19]) Churchman, t.a.p., p. 10 en 11, betwijfelt of de analytische toepassing, door hem non-systematic theory genoemd, wel tot de simulatie kan worden gerekend. Naar zijn mening valt bij de analytische toepassing de verificatie van de voorspelde „gedragswijze" buiten de simulatie. De synthetische toepassing heeft geen zin als niet aan een verificatie zou worden voldaan. Men kan daarom stellen dat in dit geval verificatie een wezenlijk onderdeel van de simulatie vormt. Churchman's twijfel berust uiteindelijk op het bij ons meer bekende onderscheid tussen een verklarende en een toegepaste theorie. Tenzij er tussen beide theorieën geen enkel verband bestaat, kan men moeilijk, in voorkomende gevallen, de een wel en de ander niet tot de simulatie rekenen.

[20]) B. Giffler, G. L. Thompson and V. Van Ness: „Numerical Experience with the Linear and Monte Carlo Algorithms for Solving Production Scheduling Problems". H. Fischer and G. L. Thompson: „Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules". Artikelen in *Industrial Scheduling*, t.a.p.

[21]) Het voorbeeld dat hier wordt gegeven veronderstelt een aantal eigenschappen van de propositielogica bekend. Voor zover van deze eigenschappen wordt gebruik gemaakt, worden zij hieronder vermeld. In de propositielogica kan iets waar of niet waar zijn, of bv. de waarden 1 en 0 aannemen. De hier gebezigde verbindingen zijn:

1. $\neg$ a, de negatie, niet a
2. a $\vee$ b, de disjunctie, a of b
3. a $\wedge$ b, de conjunctie, a en b
4. a $\supset$ b, de implicatie, a impliceert b

De verschillende waarden voor a en b en de verbindingen kunnen als volgt schematisch worden weergegeven.

a	b	$\neg$ a	$\neg$ b	a $\vee$ b	a $\wedge$ b	a $\supset$ b
1	1	0	0	1	1	1
1	0	0	1	1	0	0
0	1	1	0	1	0	1
0	0	1	1	0	0	1

[22]) A. Newell, J. C. Shaw and H. A. Simon: „A Variety of Intelligent Learning in a General Problem Solver", *Self-Organizing Systems*, ed. M. C. Yovits en S. Cameron, New York 1960, pp. 153-190.

[23]) Het werd ontleend aan A. Newell and H. A. Simon: „GPS, A Program that Simulates Human Thought", *Computer and Thought*, t.a.p., pp. 279-294.

De regels, die werden gebruikt bij het vinden van een oplossing, worden hieronder vermeld. De nummers bij de regels zijn die, welke worden gebruikt in het artikel van Newell en Simon. De pijlen geven de richting aan waarin een transformatie kan plaatsvinden.

regel 1: $a \wedge b \longrightarrow b \wedge a$
$a \vee b \longrightarrow b \vee a$
regel 2: $a \supset b \longrightarrow b \supset \neg a$
regel 5: $a \vee b \longleftrightarrow \neg(\neg a \wedge \neg b)$
regel 6: $a \supset b \longleftrightarrow \neg a \vee b$
regel 7: $a \wedge (b \vee c) \longleftrightarrow (a \wedge b) \vee (a \wedge c)$
$a \vee (b \wedge c) \longleftrightarrow (a \vee b) \wedge (a \vee c)$

regel 8: $a \wedge b \longrightarrow a$
$a \wedge b \longrightarrow b$

regel 10: $\left.\begin{matrix} a \\ b \end{matrix}\right\} \longrightarrow a \wedge b$

regel 12: $\left.\begin{matrix} a \supset b \\ b \supset c \end{matrix}\right\} \longrightarrow a \supset c$

De regels 8, 10 en 12 kunnen alleen worden toegepast op de doelstelling en de ingevoerde formule.
De operaties, zie punt B vorige paragraaf, zijn, voor zover in dit geval van belang, weergegeven in de onderstaande tabel. De tabel moet over de rij worden gelezen. Dit betekent dat de machine zeven verschillen tussen de ingevoerde formule, de sub-doelstellingen en de doelstelling kan onderscheiden en bij elk van deze verschillen de bijbehorende, met een x in de tabel aangegeven, regels kan toepassen.

Verschillen \ Regels	r1	r2	r5	r6	r7	r8	r10	r12
Voeg symbolen toe					x		x	x
Verminder het aantal symbolen					x	x		x
Verander de verbinding			x	x	x			
Verander het teken			x					
Verander het eerste teken		x	x	x				
Verander de combinaties					x			
Verander de positie	x	x						

Het probleem luidt: zet formule (1) $(a \supset \neg b) \wedge (\neg a \supset c)$ om in $\neg (\neg c \wedge b)$. De proefpersoon gaf de volgende oplossing:

(2) $(\neg a \vee \neg b) \wedge (a \vee c)$ regel 6 toegepast op het linker en rechter lid van (1)
(3) $(\neg a \vee \neg b) \wedge (\neg a \supset c)$ regel 6 toegepast op het linker lid van (1)
(4) $a \supset \neg b$ regel 8 toegepast op (1)
(5) $\neg a \vee \neg b$ regel 6 toegepast op (4)
(6) $\neg a \supset c$ regel 8 toegepast op (1)
(7) $a \vee c$ regel 6 toegepast op (6)
(8) $(\neg a \vee \neg b) \wedge (a \vee c)$ regel 10 toegepast op (5) en (7)
(9) $b \supset \neg a$ regel 2 toegepast op (4)
(10) $\neg c \supset a$ regel 2 toegepast op (6)
(11) $b \supset c$ regel 12 toegepast op (6) en (9)
(12) $\neg b \vee c$ regel 6 toegepast op (11)
(13) $\neg (b \wedge \neg c)$ regel 5 toegepast op (12)
(14) $\neg (\neg c \wedge b)$ regel 1 toegepast op (13), Q.E.D.

De machine ging als volgt te werk. Enkele van de belangrijkste stappen worden hier vermeld, de gebruikte operaties staan tussen aanhalingstekens.

Doel 1: transformeer formule 1: $(a \supset \neg b) \wedge (\neg a \supset c)$ in
formule 0: $\neg(\neg c \wedge b)$

De machine merkt op dat er in formule 1 symbolen a voorkomen en in formule 0 niet. Geconstateerd wordt het verschil „verminder het aantal termen". Dit leidt tot de formulering van doelstelling 2, in dit geval een sub-doelstelling.

Doel 2: verminder het aantal a's in formule 1.

Op grond van bepaalde heuristische principes kiest de machine uit het aantal mogelijke regels, regel 8 en wel de eerste variant. Dit leidt tot een nieuwe sub-doelstelling.

Doel 3: pas regel 8 toe op formule 1, resultaat formule 2: $a \supset \neg b$.

Nagegaan wordt nu of formule 2 kan worden getransformeerd in formule 0 en wel met behulp van het verschil „voeg symbolen toe".

Doel 4: transformeer formule 2 in formule 0.

Doel 5: voeg c toe aan formule 2; het resultaat wordt verworpen. De machine keert terug naar doel 2.

Doel 6: pas regel 8 toe op formule 1, resultaat formule 3: $\neg a \supset c$.

Doel 7: transformeer formule 3 in formule 0.

Doel 8: voeg b toe aan formule 3; het resultaat wordt verworpen.

Regel 8 levert geen resultaten, wederom op grond van heuristische principes besluit de machine door te gaan met doel 2, nu met behulp van regel 7.

Doel 9: pas regel 7 toe op formule 1.

Gekozen wordt de variant $(a \vee b) \wedge (a \vee c) \longleftrightarrow a \vee (b \wedge c)$, omdat deze het aantal symbolen vermindert en een $\wedge$ als teken heeft. Transformatie is niet direct mogelijk, maar het blijkt dat er een verschil in verbinding is in zowel het linker- als rechter lid van formule 1. Dit leidt tot een nieuwe sub-doelstelling.

Doel 10: verander de verbinding in het linker lid van formule 1 in $\vee$.

Doel 11: pas regel 6 toe op het linker lid van formule 1, resultaat formule 4: $(\neg a \vee \neg b) \wedge (\neg a \supset c)$.

Doel 12: *pas* regel 7 toe op formule 4.

Doel 13: verander verbinding in $\vee$ in het rechter lid van formule 4.

Doel 14: pas regel 6 toe op het rechter lid van formule 4, resultaat formule 5: $(\neg a \vee \neg b) \wedge (a \vee c)$.

De machine probeert nu regel 7 toe te passen op formule 5.

Doel 15: pas regel 7 toe op formule 5.

Dit blijkt niet zonder meer mogelijk omdat de tekens verschillen.

Doel 16: verander het teken in het linker lid van formule 5.

Voor de „verandering van het teken" wordt gekozen regel 6. De regel die hiervoor ook reeds werd gebruikt.

Doel 17: pas regel 6 toe op het rechter lid van formule 5 en niet op het linker lid van formule 5 omdat het rechter lid van formule 5 en het rechter lid van regel 6 met elkaar overeenstemmen; resultaat formule 6: $(\neg a \vee \neg b) \wedge (\neg a \supset c)$. Dit resultaat is gelijk aan dat gevonden in formule 4. Alle pogingen die de machine daarna nog aanwendt, aan de hand van de verschillen „verander de verbinding" en „verander het teken", leiden steeds tot reeds gevonden resultaten, waaruit tenslotte de conclusie wordt getrokken dat geen vooruitgang wordt geboekt en verdere po-

gingen worden gestaakt. Een omzetting van formules waarin de machine wel slaagt, is de volgende[24]).

Doel: transformeer formule 1: $a \wedge (\neg b \supset c)$ in

formule 0: $(c \vee b) \wedge a$.

Eerste verschil: a staat in beide formules op een andere plaats ten opzichte van het teken $\wedge$. Uit „verander positie" en regel 2 resulteert formule 2: $(\neg b \supset c) \wedge a$.
Tweede verschil: een verschil tussen $\supset$ en $\vee$ leidt via „verander de verbinding" en regel 6 tot formule 3: $(b \vee c) \wedge a$.
Derde verschil: een verschil in positie tussen b en c leidt via „verander positie" en regel 2 tot formule 0, het doel.

Duidelijk blijkt uit het voorgaande de stapsgewijze procedure die bij de probleemtransformatie wordt gebruikt. Minder duidelijk, maar van evident belang, zijn de gebruikte heuristische principes. Een groot verschil tussen pogingen van de proefpersoon en die van de rekenmachine is, dat de eerste bepaalde stappen simultaan verricht en de laatste deze achtereenvolgens doet, bv. de toepassing van regel 8 en 6. Zonder bezwaar zou ook deze simultane procedure in het programma kunnen worden opgenomen.

Het niet kunnen vinden van een oplossing moet worden gezocht in het niet ontdekken van de toepassingsmogelijkheid van bepaalde operaties. Het selectieve van het programma ten aanzien van mogelijke verschillen is in wezen ook een kwestie van heuristische principes. Wel moet men voor ogen houden dat de toepassingsmogelijkheden van deze principes uiteraard worden beperkt door de omvang van de geheugenruimte van de rekenmachine.

Uit het voorgaande blijkt duidelijk de grote overeenkomst in de pogingen van mens en machine bij het zoeken naar een oplossing. Essentieel hierbij is dat de mens evenmin als de machine langs de kortste weg het gezochte resultaat vindt. Het is juist de eigenschap van de mens om problemen via „vallen en opstaan" op te lossen die tot de ontwikkeling van de heuristische methoden en programmering heeft geleid.

---

[24]) Dit geslaagde voorbeeld werd ontleend aan, A. Newell, J. C. Shaw and H. A. Simon: „A Variety of Intelligent Learning in a General Problem Solver", *Self-Organizing Systems*, pp. 184.