Heuristieken en kunstmatige intelligentie

van den Herik, H.J.

Published in:
Wijsgerig perspectief op maatschappij en wetenschap

Publication date:
1988

Heuristieken en kunstmatige intelligentie

H.J. van den Herik

Samenvatting

Bij het oplossen van problemen zijn computers bijzonder goede hulpmiddelen gebleken. Aanvankelijk werden ze beschouwd als rekenmachines, maar spoedig brak het inzicht door dat ze ook met kennis konden manipuleren. Dit opende nieuwe perspectieven voor probleemoplossers.

Na een formele beschrijving van probleemruimte en bijbehorende toestandsruimte wordt in deze bijdrage ingegaan op de toepassing van heuristische kennis bij het oplossen van een probleem. We zien dat heuristische kennis een rol speelt (i) bij de opstelling van de kennisregels, (ii) in de besturingsstrategie en (iii) in het zoekproces. Er wordt aandacht besteed aan de mogelijkheid dat feitelijke kennis en heuristische kennis de combinatorische explosie kunnen uitstellen maar niet overmeesteren.

Het gegeven model wordt uitgewerkt voor een handelsreizigersprobleem voor vijf steden. De resultaten van twee heuristische algoritmen (de naaste-buuralgoritme en de greedy-algoritme) worden vergeleken met een optimale oplossing die verkregen is door een brute-krachtalgoritme.

Tot slot wordt betoogd dat onbewuste (of onderbewuste) heuristieken niet alleen feilbaar kunnen zijn, maar ook remmend kunnen werken op het oplos-

singsproces doordat buitenwetenschappelijke noties meespelen.

1. Inleiding

De komst van computers aan het eind van de jaren veertig betekende voor wiskundigen het beschikbaar komen van een riant hulpmiddel bij het oplossen van hun numerieke problemen. Het ligt evenwel in de aard van elke probleemoplosser dat hij na het verkrijgen van een oplossing een complexer probleem aanpakt, zodat hij opnieuw de grenzen van het mogelijke, dat in dit geval tijdgebonden is, ervaart. Omstreeks 1950 kon men dan ook in diverse wiskundige tijdschriften concepten (of algoritmen) voor berekeningen van numerieke problemen aantreffen.

Langzaam maar zeker drong toen bij sommigen het besef door dat computers behalve voor het manipuleren met getallen ook gebruikt zouden kunnen worden voor het manipuleren met kennis. Deze gedachte is door Shannon en Turing ongeveer tegelijkertijd op papier gezet. Beiden kunnen gezien worden als grondleggers van de Artificial Intelligence (AI; in het Nederlands: kunstmatige intelligentie).

Zij publiceerden de uitwerking van hun gedachten niet in wiskundige tijdschriften (informaticatijdschriften waren er nog helemaal niet), maar in filosofische tijdschriften. Hoewel beide geleerden het schaken als een voorbeeld bij uitstek zagen van niet slechts rekenen, maar vooral strategisch handelen (denken), beschreef Turing (1950) zijn denkbeelden in brede zin onder de titel *Computing machinery and intelligence*. Met name de ondertitel *Can machines think?* bracht toen heel veel pennen in beroering. De discussie is overigens heden ten dage nog steeds actueel. Shannon (1950) schreef over een directe toepassing (schaken) onder de titel *Programming a computer for playing chess*. Later deed Turing (1953) dat eveneens, maar Shannons artikel had prioriteit, ondanks een uitspraak van Turings rechterhand, I.J. Good, dat Turing de gedachte aan schaakspelende computers al in 1944 had ontwikkeld en dat hij, Good, hem van publikatie had afgehouden, omdat de algoritme hem te vanzelfsprekend leek.

2. AI-onderzoek

De eerste vier gebieden van AI-onderzoek waren: *schaken, dammen op de 64 velden (checkers)*, het automatisch bewijzen van theorema's en het automatisch vertalen. In de beginjaren dacht men dat het erom ging een groot aantal mogelijkheden te onderzoeken om daarna een *beslissing* te nemen. De vier onderzoeksgebieden werden vooral gezien als *zoekprobleem*.

Toen men halverwege de jaren zestig begon te onderkennen dat dit zoeken niet tot adequate programma's leidde, vond er een verschuiving plaats van zoektechnieken naar kennisrepresentatie. De gedachte was dat als er maar genoeg kennis van hoog niveau geïmplementeerd werd in een programma, er 'vanzelf' wel een goede zet of oplossing te voorschijn zou komen. Dit bleek echter slechts in heel beperkte mate waar te zijn; in feite gelukte het niet om het *kennisprobleem* op te lossen.

Een *combinatie* van speciale kennisrepresentaties en zoektechnieken leidde op deelgebieden (b.v. schaakeindspelen) tot goede prestaties, maar bleek voor het gehele domein niet realiseerbaar.

3. De term heuristiek

Bij het nemen van beslissingen spelen *heuristieken* een grote rol. Toen Polya de term ongeveer 40 jaar geleden op zijn colleges introduceerde, werd het begrip in feite al aarzelend toegepast in de tezelfdertijd ontstane Operation Research (OR). Het duurde evenwel nog tot de jaren zestig alvorens de term in de betekenis van ervaringsregels opdook in OR-tekstboeken; waarschijnlijk voornamelijk als gevolg van Polya's (1957) publikatie en Allen Newells veelvuldig gebruik van de term (Newell was een student van Polya in Stanford) (McCorduck 1979).

4. Probleembeschrijving

In veel gevallen kan een probleem geformuleerd worden in termen van een toestandsruimtebeschrijving. In figuur 1 zien we een geformaliseerde probleemruimte, waarbinnen operaties gedefinieerd worden.
De structuur van een probleemruimte komt in twee opzichten overeen met de structuur die we kunnen waarnemen bij het oplossen van problemen.

1. Er moet een formele definiëring plaatsvinden. Hiervoor zijn de volgende relaties van belang:
   a. gegeven probleemsituatie(s) moet(en) afgebeeld worden op begintoestand(en);
   b. gewenste situatie(s) moet(en) afgebeeld worden op eindtoestand(en) [doel(en)];
   c. verzameling toegestane operaties moet afgebeeld worden op regels in de rule-base [regelbank].
2. Het oplossen van een bepaald probleem kan be-

schouwd worden als een combinatie van:
a. bekende technieken, zoals het toepassen van logische regels, het maken van gevolgtrekkingen, etc.;
b. zoekprocedures, zoals het eerst in de diepte zoeken of het eerst in de breedte zoeken.

De data-base en de rule-base vormen te zamen de knowledge-base (kennisbank). Soms bevat de knowledge-base daarenboven een hypothesis-base om het zoekproces te besturen indien voor een achterwaartse zoekstrategie is gekozen.

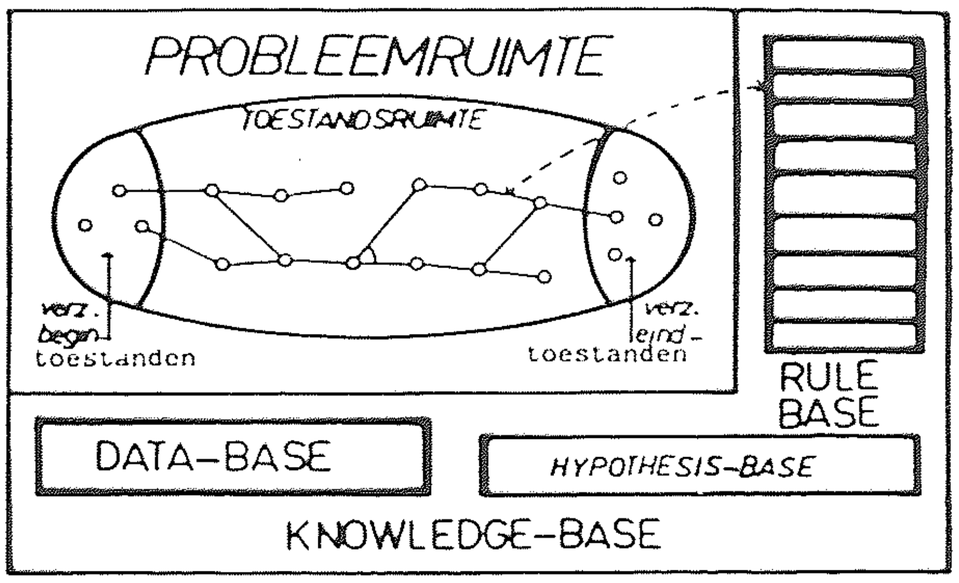
We merken nog op dat de data-base *feiten* over het domein bevat en de rule-base *regels* over het domein. Deze regels kunnen de vorm hebben van operatoren, die het zoekproces van de ene toestand in een andere doen overgaan.

5. Het gebruik van heuristieken

Heuristieken kunnen bij het oplossen van problemen op drie manieren een rol spelen (zie figuur 1):
1. heuristieken spelen een rol bij de formulering van de regels (operatoren) in de rule-base;
2. heuristieken besturen het zoekproces in de rule-base (welke regel kiezen we?);
3. heuristieken besturen het zoekproces in de toestandsruimte (welke toestand kiezen we?).

Het oplossen van AI-problemen wordt daarom ook wel gedefinieerd als een proces van heuristisch zoeken.

Figuur 1: Probleemruimte

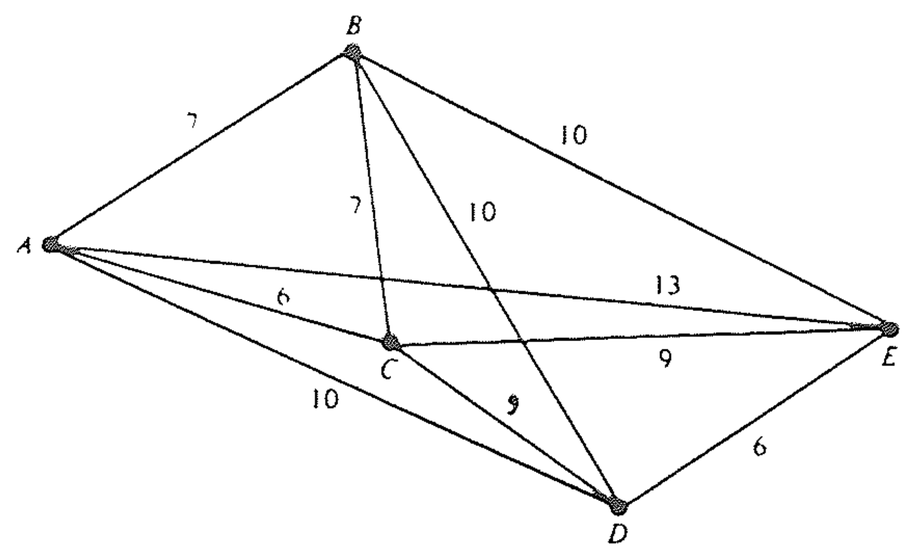


6. Het handelsreizigersprobleem

Een bekend probleem in diverse takken van wetenschap is het handelsreizigersprobleem. Wij bezien hieronder een vijf-stedenprobleem (cf. Nilsson 1980) en beschouwen twee heuristische algoritmen nadat we eerst de oplossing voor dit kleine probleem met een brute-krachttechniek hebben gevonden.

Probleem: Een handelsreiziger moet elk van de vijf steden uit figuur 2 bezoeken. Er is een weg tussen ieder tweetal steden; naast de weg staat de afstand vermeld. De handelsreiziger start in stad A. Het probleem is een route te vinden met minimale afstand zodanig dat hij elke stad slechts een keer bezoekt en ten slotte terugkeert in A.

Figuur 2: De ligging van de vijf steden



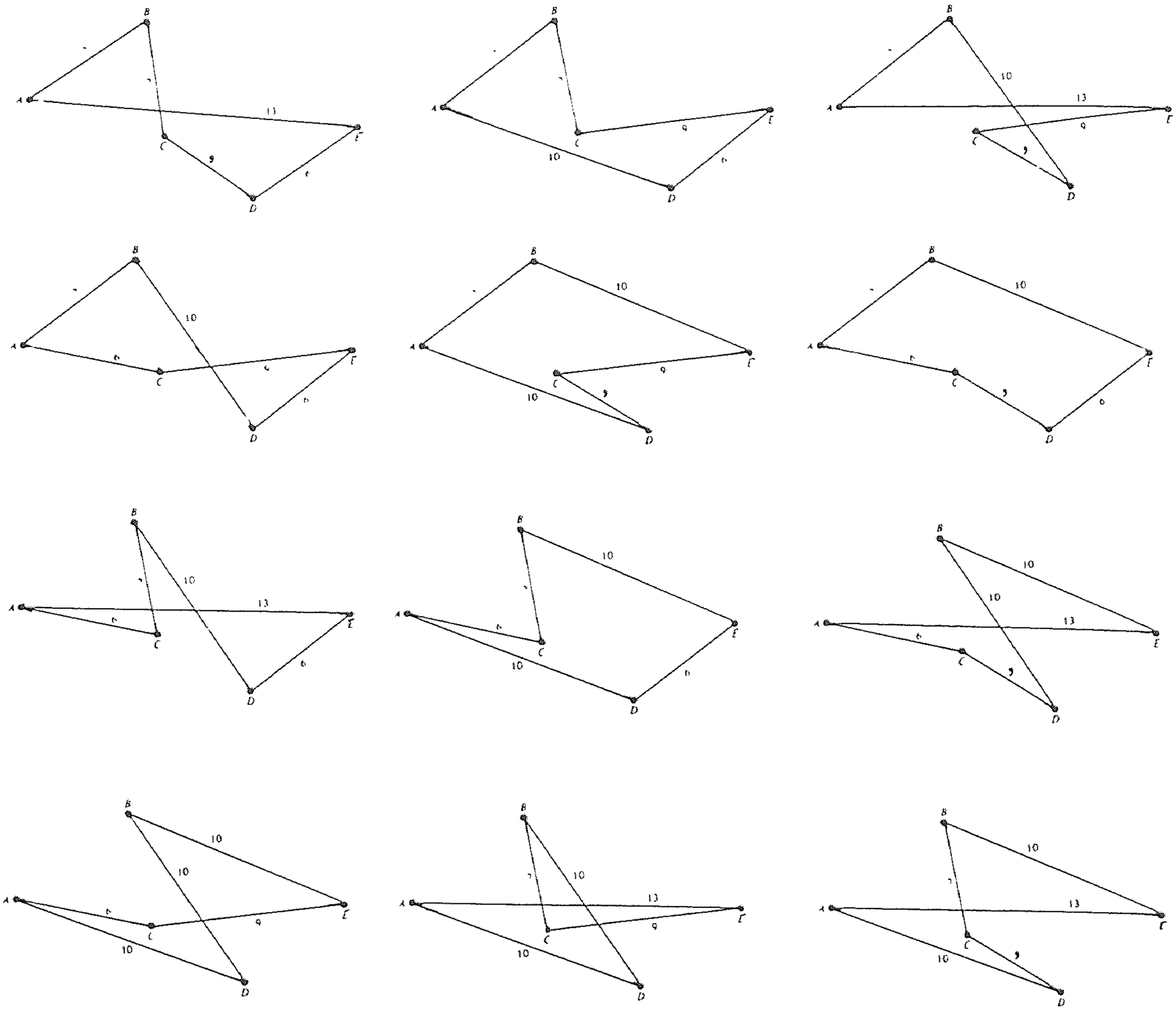
Er bestaat een eenvoudige oplossing waarbij alle mogelijke rijtjes volgens welke de steden kunnen worden bezocht, worden genoteerd. De afstanden worden per rijtje opgeteld en het rijtje met de kortste route is de gevraagde route. Als er vijf steden zijn, bestaan er slechts hoogstens 12 mogelijke lengten van routes. Het is duidelijk dat de kortste route onafhankelijk is van de startplaats, omdat de handelsreiziger hier toch weer moet terugkeren. Vanuit de eerste stad is de keuzevrijheid 4, daarna 3, vervolgens 2 en ten slotte 1 (geen keuze). Dit betekent 4 x 3 x 2 x 1 = 24 verschillende routes (4! = 1 x 2 x 3 x 4). Omdat de lengte van een route onafhankelijk is van de gevolgde richting kunnen we stellen dat er slechts 24: 2 = 12 routes zijn (12 = (5–1)!/2). In het algemeen geldt voor het N-stedenprobleem dat er (N–1)!/2 routes bekeken moeten worden. Voor het vijf-stedenprobleem zijn de routes opgesomd in figuur 3.

Het probleem valt gemakkelijk op te lossen met het zoeken in een boom, zoals aangegeven in figuur 4. Een oplossing luidt A B D E C A, met lengte 38.

7. Exponentiële groei

De gegeven oplossing is gevonden met de methode van uitputtende opsomming (d.w.z. door gewoon *alle* mogelijkheden te onderzoeken). De genoemde methode *faalt* als het aantal steden toeneemt. Voor

Figuur 3: Alle routes voor het vijf-stedenprobleem



Figuur 4: Een zoekboom voor het vijf-stedenprobleem

N-steden zouden (N–1)!/2 routes onderzocht moeten worden; we zeggen dat dit een probleem is van de Orde(N!). Verder zij vermeld dat 10! = 3.628.000 en 20! = 2.432.902.008.176.640.000 (~ $2{,}4\ 10^{18}$).

Problemen kunnen geordend worden volgens een tijdshiërarchie (tijdscomplexiteit) die benodigd is om het 'slechtste geval' (worst case) op te lossen met gebruikmaking van de theoretisch best bekende methode. We kennen de volgende groeionderscheiding voor het toenemen van de omvang van een probleem:

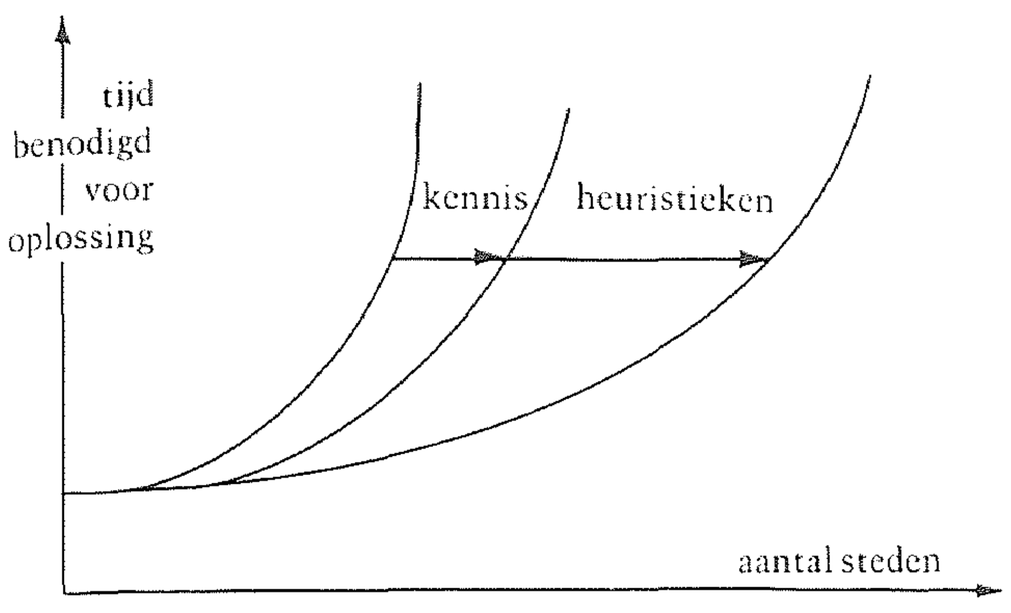
- lineaire groei;
- polynomiale groei;
- exponentiële groei.

De brute-krachtmethode is van de orde N! en behoort tot de exponentiële groei. Er zijn betere methoden, maar we moeten toch alvast opmerken dat de theoretisch best bekende methoden niet in staat zijn de *beste*

oplossing te vinden zonder met de exponentiële explosie geconfronteerd te worden.

AI-inspanningen trachten de groei van tijd-versus-probleemomvang-curve (zie figuur 5) zo veel mogelijk te niet te doen door:
- kennis over het probleemgebied te gebruiken;
- heuristieken te gebruiken.

Figuur 5: De exponentiële groei van het N-stedenprobleem



8. Het gebruik van kennis

In de boomzoekprocedure zoals die in figuur 4 is weergegeven kunnen we de zogeheten branch-and-bound-techniek toepassen. Dit wil zeggen dat we de volgende *kennis* toevoegen: 'Als een partieel pad al langer is dan het kortste pad tot dan toe gevonden, behoeft er niet verder gezocht te worden'. Gebruik van deze kennis resulteert in de volgende algoritme:

1. Begin met een volledig pad te genereren (noem dit 'het kortste pad tot dusver').
2. Ieder ander pad dat korter is zal genoemd worden: 'het kortste pad tot dusver'.
3. Staak het onderzoek van een pad zo snel als z'n partiële lengte groter is dan 'het kortste pad tot dusver'.

Dit is een efficiëntere methode dan de brute-kracht-methode, maar toch ontkomt deze toevoeging van kennis niet aan de exponentiële explosie. De tijdscomplexiteit is van de Orde $(1, 26)^N$. Om dit verder te verbeteren bezien we nu twee heuristieken die we achtereenvolgens in algoritmevorm presenteren.

9. Het gebruik van heuristieken

Er vallen twee typen van heuristieken te onderscheiden:

(i) de algemene heuristieken;
(ii) de domeingebonden heuristieken.

In ons geval hebben we het probleem teruggebracht tot een zoekprobleem dat in wezen slechts op een zoekproces berust. Wij bezien daarom twee algemene heuristieken, te weten de *naaste-buurheuristiek* en de *greedy-heuristiek*. Zij kunnen in veel problemen toegepast worden.

9.1 *De naaste-buurheuristiek* – De naaste-buurheuristiek gaat ervan uit dat het gunstig is om telkens de kortste afstand af te leggen naar een volgende toestand in de toestandsruimte. Voor het handelsreizigersprobleem betekent dit dat telkens de dichtstbijzijnde stad bezocht wordt en dat aan het eind de afstand tot de beginstad overbrugd dient te worden. Deze heuristiek ziet er buitengewoon plausibel uit (kenmerk van heuristieken!).

Echte probleemoplossers getroosten zich soms echter wel eens een opoffering om later meer winst binnen te halen en dit element mist de naaste-buurheuristiek (in honkbaltermen: een opofferingsstoot).

In algoritmevorm kunnen we de heuristiek als volgt formuleren:

1. *Kies* een startplaats.
2. *Kies* een stad, die het dichtst bij de laatst bezochte stad ligt en die nog niet eerder bezocht is.
   Ga naar deze stad.
3. Herhaal stap 2 totdat alle steden bezocht zijn.
4. Ga naar de startplaats.

De tijdscomplexiteit is van de Orde (N^2) (polynomiale groei), derhalve hebben we te maken met een aanzienlijke verbetering. In de literatuur hebben verschillende auteurs zich beziggehouden met het geven van een bovengrens van de fout die bij toepassingen van deze algoritme gemaakt kan worden. In paragraaf 10 werken we deze algoritme uit voor het vijf-stedenprobleem.

9.2 *De greedy-heuristiek* – De greedy-heuristiek heeft als uitgangspunt dat het gunstig is als zo veel mogelijk korte afstanden in het uiteindelijke pad zijn opgenomen. Voor het handelsreizigersprobleem betekent dit dat alle afstanden in een lijst worden gezet en er door een algoritme onderzocht wordt of de betreffende afstand in een oplossing zou passen. Het formuleren van de precieze voorwaarden vereist nog enige nauwkeurigheid. Hieronder presenteren we een ruwe versie van de greedy-algoritme, die voor ons doel voldoende geschikt is.

1. Plaats alle wegen in een lijst en order ze van groot naar klein.
2. *Kies* een weg met de kortste afstand die ten minste

één nog niet bezochte plaats verbindt.
3. Herhaal stap 2 totdat alle steden bezocht zijn.
4. Verbindt twee uiteinden van het minimale skelet met elkaar en ga na of dit een mogelijke oplossing is.

Ook hier hebben we te maken met een aanzienlijke verbetering die tot een polynomiale groei leidt. Aan de andere kant speelt het verschijnsel van de 'opofferingsstoot' in deze algoritme een grotere rol dan bij de naaste-buuralgoritme.

10. De werking van de heuristieken

Hoe eenvoudig het vijf-stedenprobleem ook mag zijn en hoe vanzelfsprekend de heuristieken ook mogen zijn, enig inzicht in de werking van heuristieken wordt pas echt verkregen door er mee om te gaan. In deze paragraaf gaan we daarom stap voor stap de in 9.1 en 9.2 beschreven algoritmen uitvoeren. Daartoe inventariseren we eerst de beschikbare afstanden.

AC	6	AB	7	CD	9	BE	10	AE	13
DE	6	BC	7	CE	9	AD	10		
						BD	10		

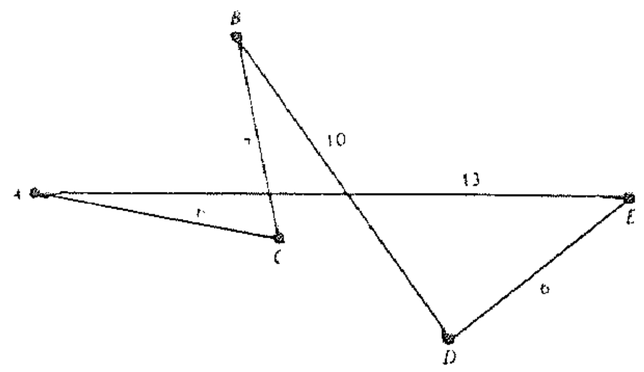
Uit deze lijst zien we dat er diverse afstanden zijn met gelijke lengte. Dit betekent dat het kan gebeuren dat er tijdens het zoekproces beslissingen genomen dienen te worden. De in 9.1 en 9.2 gegeven algoritmen hebben betrekking op heuristieken die het zoekproces in de rule-base (het zoeken naar operatoren) besturen (cf. paragraaf 5, item 2). Bij het kiezen tussen twee 'gelijke' mogelijkheden kunnen we heuristieken introduceren die het zoekproces in de toestandsruimte besturen (cf. paragraaf 5, item 3). Meestal wordt dit gedaan in de vorm van evaluatiefuncties (b.v. voor een schaakstelling). Bij het vijf-stedenprobleem kunnen we bij gebrek aan nadere informatie als heuristiek hanteren:

a. de alfabetische volgorde (zonodig lexicografisch);
b. de omgekeerde alfabetische volgorde;
c. afwisselend de alfabetische en omgekeerd alfabetische volgorde;
d. de volgorde zoals in de bovenstaande lijst.

10.1 *Het resultaat van de naaste-buurheuristiek* – Hieronder volgt de uitwerking van de naaste-buuralgoritme, zoals die gegeven is in 9.1, met als extra heuristiek de alfabetische volgorde van de plaatsen in de lijst (heuristiek a).

(i) We kiezen als startplaats: A.
Daarna wijst het algoritme ons het volgende pad (zie figuur 6):

Figuur 6

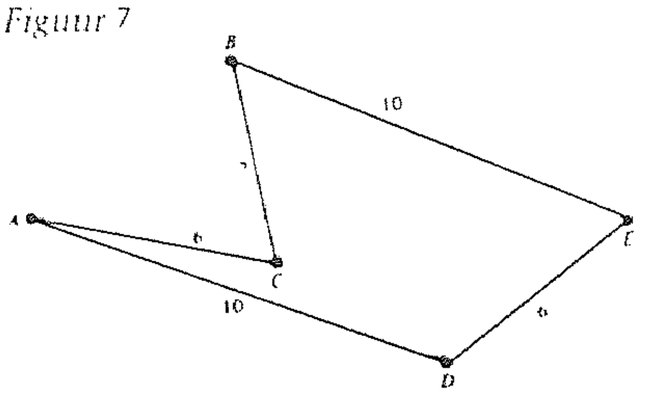


AC	6
CB	7
BD	10
DE	6
EA	13

Totale pad 42

(ii) In plaats van heuristiek a hadden we ook heuristiek d kunnen kiezen. Met startplaats A verkrijgen we dan de volgende oplossing (zie figuur 7):

Figuur 7



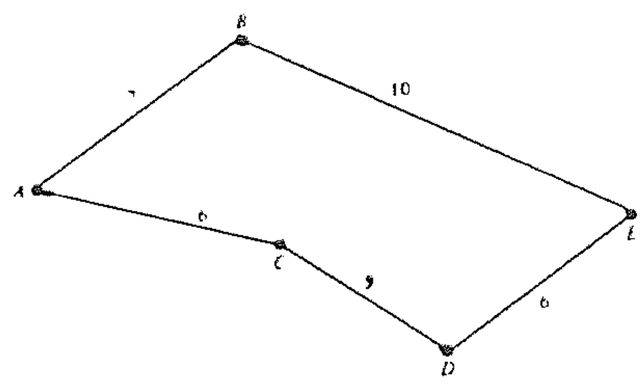
AC	6
CB	7
BE	10
ED	6
DA	10

Totale pad 39

Het is duidelijk dat een goede keuze van de heuristiek belangrijk is voor de kwaliteit van de oplossing van het probleem. Bij de brute-krachtmethode is de startplaats onbelangrijk, de (een) optimale oplossing wordt altijd gevonden. Bij de naaste-buuralgoritme speelt de keuze van de startplaats echter wel een rol zoals we hieronder zien.

(iii) We kiezen C als startplaats en heuristiek D. Dit leidt tot de volgende oplossing (zie figuur 8):

Figuur 8



CA	6
AB	7
BE	10
ED	6
DC	9

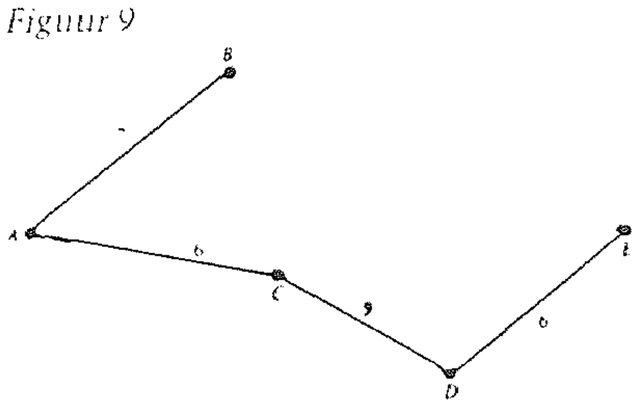
Totale pad 38

Een conclusie die uit deze drie voorbeelden getrokken kan worden is dat toepassing van de naaste-buurheuristiek nooit garandeert dat een optimale oplossing gevonden wordt. Bovendien blijkt, dat hoewel de kortste route onafhankelijk is van de startplaats (zie paragraaf 6), de keuze van een startplaats bij heuristische algoritmen wel degelijk van belang is.

10.2 *Het resultaat van de greedy-heuristiek* – Bij de uitwerking van de greedy-algoritme gebruiken we in het eerste voorbeeld heuristiek a.

(i) We vinden dan de onderstaande oplossing. In figuur 9 hebben we het gevonden minimale skelet weergegeven:

Figuur 9

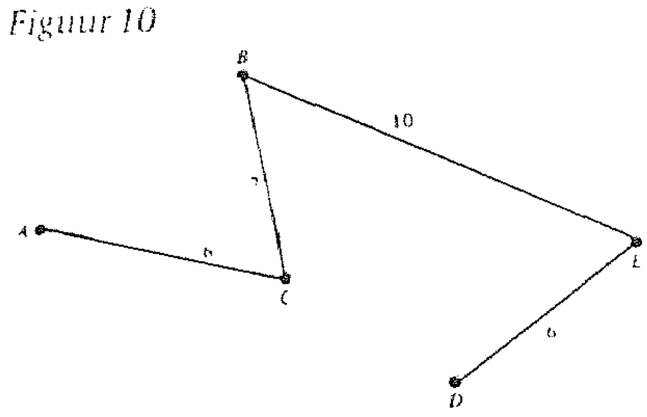


AC	6
DE	6
AB	7
CD	9
BE	10

Totale pad 38

(ii) In plaats van heuristiek a hadden we ook heuristiek b kunnen nemen. In figuur 10 wordt het resulterende minimale skelet gegeven. De oplossing verloopt dan als volgt:

Figuur 10



DE	6
AC	6
BC	7
BE	10
AD	10

Totale pad 39

11. Voorlopige conclusies

Op basis van de gegeven voorbeelden kunnen er niet veel conclusies getrokken worden. AI-onderzoekers zouden het liefst zien dat welgekozen heuristieken een perfect resultaat zouden opleveren. Dit is, zoals we gezien hebben, zeker niet altijd het geval. Toch hebben heuristieken grote voordelen:

- ze verkleinen de zoekruimte aanzienlijk;
- ze zorgen er niet zelden voor dat een oplossing binnen een acceptabele tijd wordt gevonden.

Om tot een verantwoorde keuze van heuristieken te komen, zijn de volgende vragen relevant:

- Wat is de oorsprong van een heuristiek?
- Wat is de kracht van een heuristiek?
- Wat zijn de begrenzingen van een heuristische zoekruimte?

Niet zelden worden heuristieken door een expert onbewust gebruikt. Meestal wordt dit als voorbeeld aangehaald voor de belangrijke rol die heuristieken spelen in het redeneerproces van een expert. We betogen tot slot dat onbewuste (of onderbewuste) heuristieken ook remmend kunnen werken op het oplossingsprocédé van een expert.

Meestal vormen heuristieken een goede hulp, maar soms is het nog lastiger hun tegenwerking te overwinnen. Een heuristiek die zich eenmaal in het hoofd heeft vastgezet blijkt vaak niet meer zo flexibel te zijn, juist doordat daar ook buitenwetenschappelijke noties meespelen. Schaakgrootmeesters zullen achteraf nooit precies kunnen verklaren waarom zij een bepaalde zet niet gevonden hebben.

Referenties

McCorduck, P., *Machines who think*. W.H. Freeman and Company, San Francisco 1979.

Nilsson, N.J., *Principles of artificial intelligence*. Tioga Publishing Company, Palo Alto, California 1980.

Polya, G., *How to solve it*. Princeton University Press, Princeton, New Jersey. In 1957 uitgegeven door Double day and Company. Garden City, New York 1945.

Shannon, C.E., Programming a computer for playing chess. *Philosophical Magazine*, 41, 1950, p. 256-275.

Turing, A.M., Computing machinery and intelligence. *Mind*, 59, 1950, p. 430-460.

Turing, A.M., Digital computers applied to games. *Faster than thought* (ed. B.V. Bowden). Sir Isaac Pitman, Londen 1953.

H.J. van den Herik (1947) studeerde wiskunde aan de Vrije Universiteit te Amsterdam. Thans is hij gewoon hoogleraar Informatica aan de Rijksuniversiteit Limburg en bijzonder hoogleraar aan de Juridische Faculteit van de Rijksuniversiteit Leiden. Sedert 1983 is hij Editor in Chief van het *ICCA Journal*, die is erkend als bron door het Institute for Scientific Information. Sinds 1987 is hij vice-voorzitter van de NVKI (Ned. Ver. voor Kunstmatige Intelligentie).