**Stichting Mathematisch Centrum**

**en**

**Instituut voor Actuariaat en Econometrie**

S 385 (SB 3)

# HET HANDELSREIZIGERSPROBLEEM

Een

Literatuuronderzoek

door

R. Tijdeman

scriptie besliskunde

o.l.v. Prof.dr. G. de Leve

juni 1967

INHOUD

# HOOFDSTUK 1: EEN OVERZICHT

## §1. Probleemstelling, verband met andere problemen

Litt. R.L. Ackoff, Progress in Operations Research, [1].
M.M. Flood, The Traveling Salesman Problem, [15].

Een oud probleem is de vraag hoe een paard over een schaakbord moet springen om alle velden een en slechts eenmaal te bezoeken. Oplossingen en oplossingstechnieken hiervoor werden gegeven door beroemde wiskundigen als Euler, De Moivre, Van der Monde en Legendre [2, blz. 122-132].

Omstreeks 1850 hield Sir William Hamilton zich bezig met de vraag of het mogelijk is om langs de zijden lopend alle hoekpunten van een regelmatig twintigvlak te bezoeken zonder een hoekpunt meerdere keren in de route op te nemen [2, blz. 189-192].

Het is duidelijk dat beide problemen speciale gevallen zijn van het probleem de kortste weg te vinden langs een aantal te bezoeken plaatsen, waarbij de afstand tussen elk paar punten gegeven of te berekenen is. (Voor het paard b.v. is de afstand van twee naast elkaar gelegen velden gelijk aan drie sprongen). Het is daarom verwonderlijk dat pas omstreeks 1935 de algemene formulering voor het eerst in de literatuur verschijnt:

Het handelsreizigersprobleem:

Een koopman wenst van huis vertrekkend n-1 gegeven steden te bezoeken en tenslotte weer huiswaarts te keren. Hij kent de afstand tussen elk paar steden en vraagt zich af hoe hij moet rijden om de totaal af te leggen afstand zo klein mogelijk te houden.

Het is duidelijk dat het geen verschil maakt of we "afstand" door "kosten" vervangen. Ook hoeft de afstand $C_{ij}$ van stad i naar stad j niet gelijk te zijn aan de afstand $C_{ji}$. (We geven een stad aan met zijn nummer). Mocht het de koopman niet interesseren in welke stad de tocht eindigt, dan lossen we bovenstaand probleem op, nadat we de afstand van elke stad tot zijn woonplaats op 0 eenheden gesteld hebben. Iets dergelijks kunnen we voor het startpunt doen. In de oplossing laten we dan

de laatste resp. eerste reis weg. In het algemeen noemen we een gesloten tocht langs alle steden een ronde of route.

Anders geformuleerd wordt in het handelsreizigersprobleem gevraagd om van een gegeven kostenmatrix $C = (C_{ij})_{i,j=1}^{n}$ een cyclische permutatie $(1\ i_2 \ldots i_n)$ van de getallen 1, 2, ..., n zo te bepalen dat $C_{1i_2} + C_{i_2i_3} + \ldots + C_{i_n1}$ minimaal is. We zeggen daarom in plaats van ronde ook wel cyclische permutatie.

Laten we in de laatste formulering de eis vallen, dat de permutatie cyclisch is, dan krijgen we het z.g. (persoons)toekenningsprobleem, [42], dat men in de volgende situatie krijgt:
Er zijn n mensen beschikbaar om n taken te vervullen. Verder is een $n \times n$-kostenmatrix gegeven, waarin de elementen de kosten aangeven die verbonden zijn aan het vervullen van een bepaalde taak door een bepaalde persoon. Gevraagd wordt de kosten te minimalizeren als iedereen één taak moet uitvoeren.

Mathematisch equivalent hiermee is het volgende transportprobleem: n schepen liggen in n gegeven exporthavens, ieder met dezelfde goederen aan boord. Deze goederen moeten naar n andere havens vervoerd worden, waarbij de kosten om van exporthaven i naar importhaven j te varen gegeven zijn in het element $C_{ij}$ van de $n \times n$-matrix C. Ook in dit geval wordt gevraagd de routes zo te bepalen dat de kosten minimaal zijn.

Deze problemen zijn speciale gevallen van het distributieprobleem:

$$\text{minimalizeer} \quad z = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}\, C_{ij}$$

$$\text{onder} \quad \sum_{i=1}^{m} x_{ij} = r_j,\ j = 1,2,\ldots,n; \qquad \sum_{j=1}^{n} x_{ij} = s_i,\ i = 1,2,\ldots,m;$$

$$x_{ij} \geq 0 \quad i = 1,2,\ldots,m;\ j = 1,2,\ldots,n,$$

waarbij $m,n,r_j,s_i$ en $C_{ij}$ gegeven gehele getallen zijn zodat $\sum_{j=1}^{n} r_j = \sum_{i=1}^{n} s_i$.

Dit probleem is het mathematisch model voor verschillende praktijkgevallen. Het is vanzelfsprekend dat het distributieprobleem een bijzonder geval is van het integer L.P. probleem. Het toekenninsprobleem ontstaat uit het distributieprobleem door $m = n, r_1 = r_2 = \ldots = r_n = s_1$

$= s_2 = \ldots = s_m = 1$ te stellen.
De formulering die we dan krijgen is n.l. equivalent met de volgende:

$$\text{minimalizeer} \quad z = \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} C_{ij}$$

$$\text{onder} \sum_{i=1}^{n} x_{ij} = \sum_{j=1}^{n} x_{ij} = 1, \; x_{ij} = 0 \text{ of } 1. \quad i,j = 1,2\ldots,n.$$

Helaas geldt een dergelijke equivalentie niet voor de analoge formulering van het handelsreizigerprobleem:

$$\text{minimalizeer} \quad z = \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} C_{ij}$$

$$\text{onder} \sum_{i=1}^{n} x_{ij} = \sum_{j=1}^{n} x_{ij} = 1, \; x_{ij} = 0 \text{ of } 1, \; i,j = 1,2,\ldots,n,$$

$$x_{i_1 i_2} + x_{i_2 i_3} + \ldots x_{i_{r-1} i_r} + x_{i_r i_1} \leq r-1, r=2,3,\ldots,\left[\frac{n}{2}\right]$$

als $(i_1 i_2 \ldots i_r)$ een r-cykel is uit de verzameling $\{1,2,\ldots.n\}$, en $x_{ii} = 0, \quad i = 1,2,\ldots,n.$

Dat we de voorwaarde $x_{ij} = 0$ of 1 niet mogen vervangen door $x_{ij} \geq 0$ is een aanwijzing voor het verschil in moeilijkheid bij het oplossen van beide problemen [15, p. 66].

Onafhankelijk van elkaar hebben Bellman [5], Gonzalez [18] en Held en Karp [20] een formulering ontwikkeld waarop de dynamische programmeringstechniek direct kan worden toegepast. In §7 zullen we hier verder op in gaan.

Een niet-essentiele uitbreiding van het handelsreizigersprobleem is het geval waarin een reiziger meerdere malen in een stad mag terugkeren. Een minder triviale uitbreiding wordt bestudeerd in [27]: Minimalizeer de totale afstand die een reiziger moet afleggen, die vertrekt uit stad 1, de steden 2,3,...,n precies eenmaal bezoekt, maar na maximaal p steden bezocht te hebben eerst naar 1 moet terugkeren alvorens zijn tocht te mogen vervolgen. Het aantal keren t dat de reiziger naar stad 1 terugkeert mag hierbij vast zijn of variabel.
Voor $p = n$, $t = 1$ krijgen we ons oude probleem terug. Een andere beper-

king zou kunnen zijn dat de totale afstand tussen twee opeenvolgende bezoeken aan 1 aan een maximum gebonden is.

Een andere generalisatie wordt beschreven in [39]:
Stel dat een eindig netwerk met knopen $K_1, K_2, \ldots, K_n$ gegeven is en dat gevraagd wordt de kortste verbinding te bepalen tussen $K_1$ en $K_n$, waarbij geeist wordt dat de tocht gaat door de steden $K_2, K_3, \ldots, K_i$.
Voor i = n-1 zijn we weer terug bij het handelsreizigersprobleem.
Voor i = 0 krijgen we het z.g. kortste weg probleem, dat vraagt in een gegeven (eindig) netwerk de kortste verbinding tussen twee knopen te bepalen, zie [32].

Er bestaat ook een langste weg probleem. Hierbij wordt gevraagd de langste weg te vinden tussen twee knopen in een gegeven (eindig) netwerk, waarbij de weg niet tweemaal over dezelfde knoop mag gaan.
In [19] tonen Hardgrave en Newhauser aan dat het handelsreizigersprobleem een speciaal geval is van het langste weg probleem. Zij stellen dat dit eventueel een manier zou kunnen zijn om een efficiënt algorithme voor het handelsreizigersprobleem op te stellen. Terecht stelt S.N. Narahari Pandit in [29] hier tegenover dat dit zeer onwaarschijnlijk is, daar beide problemen zo verwant zijn dat ze dezelfde oplossingsmoeilijkheden bezitten.

Ordeningsproblemen (sequencing problems) komen vaak in de besliskunde voor [1, blz. 295-326]: Stel dat een aantal karweien gedaan moet worden en dat daarvoor een aantal machines ter beschikking staat. We stellen de kosten verbonden aan het volbrengen van taak i door machine j weer bekend. Gevraagd wordt onder de laagste kosten de taken uit te voeren.

Er kunnen allerlei bijvoorwaarden zijn. Zo is bijvoorbeeld het volgende ordeningsprobleem equivalent met het handelsreizigersprobleem: Een machine heeft n taken te verrichten, $T_1, T_2, \ldots, T_n$ en begint daarna weer opnieuw. De kosten om de machine van de eindtoestand van taak i te brengen in de begintoestand van taak j geven we aan met $C_{ij}$. Het probleem is nu de totale veranderingskosten te minimalizeren. Het is daarbij duidelijk dat een taak correspondeert met een stad en $C_{ij}$ met de afstand van stad i tot stad j. Bij de oplossing van het probleem kun-

nen we echter vaak gebruik maken van het feit dat bepaalde taken beslist vóór andere taken gedaan moeten worden. In [38] wordt uitgewerkt hoe soms door middel van een klasseindeling het probleem tot een veel eenvoudiger handelsreizigersprobleem kan worden gereduceerd. Ook andere ordeningsproblemen kunnen als handelsreizigersprobleem geformuleerd worden [31,43].

Het volgende ordeningsprobleem is een speciaal geval van het handelsreizigersprobleem. In 1964 werd hiervoor een goede oplossingstechniek door P.C. Gilmore en R.E. Gomory gegeven [16,17]:
Er is een machine (b.v. een fornuis) die n taken moet verrichten. De toestand van de machine wordt volledig gekarakterizeerd door een reële variabele x (de temperatuur van het fornuis). Om taak $T_i$ te kunnen beginnen moet de machine in toestand $A_i$ verkeren, terwijl de toestandsvariabele na afloop van deze taak $B_i$ is. De kosten om van $B_i$ naar $A_j$ om te schakelen, $C_{ij}$, worden gegeven door

$$\begin{cases} C_{ij} = \int_{B_i}^{A_j} f(x)dx & \text{als } A_j \geq B_i \\ C_{ij} = \int_{A_j}^{B_i} g(x)dx & \text{als } A_j < B_i, \end{cases}$$

waarbij f,g integreerbare functies zijn zó dat $f + g \geq 0$.
Het probleem de kosten te minimalizeren kan nu, zoals Gilmore en Gomory bewezen hebben, opgelost worden in $O(n^2)$ enkelvoudige stappen.

Zoals er vele problemen qua formulering verwant zijn met het handelsreizigersprobleem, zo zijn ook vele problemen dat qua oplossingstechniek [9,20,25]. Maar zonder aanspraak te maken op enige volledigheid zullen we hier onze rondreis beeindigen en terugkeren naar gebieden waar tenminste de afstand tussen elk paar steden gegeven is.

## §2. Overzicht van de bestaande oplossingstechnieken

We zullen in het vervolg van een (toegelaten) oplossing van het handelsreizigersprobleem spreken als we een cyclische permutatie van de steden aangeven waarlangs de reiziger zijn tocht maakt. De route geven

we aan door een rijtje steden: $i_1 i_2 \ldots i_n i_1$ of $(i_1 i_2 \ldots i_n)$.
De ronde $(i_n i_{n-1} \ldots i_1)$ zullen we de inverse ronde noemen. De oplossing met de kleinste totale afstand of lengte noemen we de beste oplossing, de optimale oplossing of kortweg de oplossing.
Een oplossing die een lengte heeft die niet veel van de beste oplossing verschilt noemen we een goede oplossing. De kortste weg tussen twee steden noemen we een schakel, een rij opeenvolgende schakels een verbinding.

Er schijnen mensen te zijn die menen dat het handelsreizigersprobleem onoplosbaar is [26]. Dit is gelukkig onjuist. Er zijn zelfs maar eindig veel mogelijke oplossingen, om precies te zijn (n-1)! mogelijkheden in het asymmetrische, (n-1)!/2 mogelijkheden in het symmetrische geval. De moeilijkheid is een efficiënte methode te ontwikkelen om de beste permutatie te vinden. Het is duidelijk dat de komst van de computer van zeer groot belang is geweest voor de manier waarop men het probleem heeft aangepakt.

Vóórdat de computer als hulpmiddel werd toegepast, waren algemene (vaak meetkundige) stellingen van belang. Een voorbeeld daarvan is de stelling die Flood in [15] geeft:
In de twee-dimensionale Euclidische ruimte doorsnijdt de beste oplossing zichzelf niet.
Een direct gevolg van deze stelling is: vormen de steden de hoekpunten van een convexe veelhoek, dan vormen de zijden van die veelhoek juist de optimale ronde. Gebruik makend van dit soort stellingen heeft Barachet [3] een methode ontwikkeld die een goede route geeft. Optimaliteit is niet gegarandeerd en de methode is ook niet erg handelbaar, maar ze geeft niettemin een goed beeld van de aard van het probleem. In 1962 nog werd de methode door Gonzalez aangewezen als hulpmiddel voor de door hem voorgestelde methode.

Barachet's artikel verscheen in 1957. Het was niet de eerste keer, dat een oplossingsmethode gegeven werd. Reeds in 1954 hadden Dantzig, Fulkerson en Johnson [10] van een probleem van 49 steden in de Verenigde Staten een oplossing gegeven waarvan ze bewezen dat deze optimaal is. Een bewonderenswaardige prestatie, al is het dan een betrekkelijk

eenvoudig probleem voor 49 steden. De gevolgde oplossingsmethode is gebaseerd op de integer L.P. formulering zoals deze gegeven is in §1. Ze beperken zich tot het symmetrische geval.

Uitgaande van een toegelaten route en een klein aantal lineaire vergelijkingen en ongelijkheden, waaraan door alle cyclische permutaties voldaan is, berekenen ze een nieuwe basisoplossing met de Simplexmethode. In het geval de nieuwe oplossing niet toegelaten is, voegen we nieuwe lineaire eisen toe, zoals dit in principe bij het integer L.P. probleem wordt toegepast. Tenslotte vinden we zo een goede oplossing. Zouden we beslist de beste oplossing willen vinden dan staan ons twee wegen open. We bepalen de (hopelijk beperkte hoeveelheid) oplossingen die een kortere lengte zouden kunnen hebben en onderzoeken deze hetzij met combinatorische argumenten hetzij door al deze lengtes uit te rekenen en te vergelijken.

I. Heller heeft bovenstaande techniek theoretisch onderzocht en een gedeeltelijke rechtvaardiging gegeven voor de voorgestelde toe te voegen ongelijkheden [21].

Anderen die oplossingstechnieken ontwierpen uitgaande van de integer L.P. formulering zijn Miller, Tucker en Zemlin in [27] en Mudrov in [28].

Naast deze technieken die een algemeen uitgangspunt hebben zijn er ook enkele methoden ontwikkeld die alleen van de specifieke eigenschappen van het handelsreizigersprobleem gebruik maken:

In 1958 gaf de Nederlander Croes in [7] een methode aan die verwant is aan de methode van Dantzig e.a. In de eerste stap bepaalde hij een geschikte startroute. Vervolgens stelde hij voor deze door inversies (omkeringen van gedeelten van de cyclische permutatie) te verbeteren en tenslotte gaf hij een zeer ingewikkelde methode aan om de optimale route te bepalen en te bewijzen dat deze optimaal is.

In 1959 verscheen een rapport van Dacey [8], waarin hij een heuristische methode geeft om een goede oplossing te krijgen. Voor een verslag van zijn resultaten, zie [8 , abstract].

Karg en Thompson publiceerden in 1964 een artikel in Management Science, [23],waarin ze eveneens een heuristische methode aangeven om

een goede oplossing te berekenen. Zij hebben hun methode op de computer getest en kwamen tot goede resultaten. Hierop hopen we in §10 nog in te gaan.

De na Barachet genoemde methoden samenvattend komen we tot de conclusie dat er in de jaren 1954-1964 een aantal technieken is ontwikkeld waarbij we met eenvoudige, te programmeren methoden een goede oplossing krijgen, maar waarbij het veel moeite kost en ook vaak zeer ingewikkeld is de beste oplossing te vinden en te bewijzen dat deze oplossing ook optimaal is. Nadat we in §3 wat dieper op de methode van Barachet zijn ingegaan, zullen we in §§4, 5 en 6 een numeriek voorbeeld geven van de methode van respectievelijk Dantzig, Fulkerson en Johnson, Croes en Karg en Thompson.

Een geheel nieuwe ontwikkeling vond plaats omstreeks 1962. In dat jaar verschenen drie artikelen, [5], [18], [20], die het handelsreizigersprobleem aanpakken met een methode, die gebaseerd is op de theorie van de dynamische programmering. Hoewel Bellman in 1960 een voordracht had gehouden tijdens een Symposium van de American Mathematical Society, [4], en de andere auteurs bij hun publicatie dit werk kenden, schijnt de methode toch onafhankelijk van elkaar door hen ontwikkeld te zijn.

Deze methode is, zoals we zullen zien, geschikt om problemen van ten hoogste 17 steden op te lossen, maar heeft de belangrijke eigenschap dat ze het algemene probleem aanpakt en volledig geschikt is om geprogrammeerd te worden. Een programma wordt b.v. gegeven in [18], blz. 86-95. Voor een groter aantal steden hebben Held en Karp in [20] aangegeven, hoe ze een goede oplossing voor dat probleem berekenen.

De optimale oplossing voor een klein probleem wordt als volgt gevonden: Bereken voor $i,j = 2,3,\ldots,n$ de lengte van de verbinding $1ji$. Bereken hieruit de kortste verbinding van 1 naar i die gaat langs de steden j en k, dus het minimum van de afstanden van $1jki$ en $1kji$. Bereken vervolgens voor $i,j,k,l = 2,3,\ldots,n$, alle onderling verschillend, de kortste verbinding van stad 1 naar stad i langs de steden j,k en l. Zet dit proces voort totdat tenslotte de kortste verbinding van 1 naar 1 gevonden wordt, die voert langs de steden $2,3,\ldots,n$.

Ter verkorting kunnen we in het symmetrische geval ongeveer op de helft ophouden en de gesloten permutatie zoeken die de kleinste totale afstand heeft. Is b.v. n even, dan beschouwen we daartoe alle verbindingen $1i_2i_3 \ldots i_{\frac{n}{2}} i_{\frac{n}{2}+1}$ en $1i_n i_{n-1} \ldots i_{\frac{n}{2}+1}$ met $i_j \neq i_k$ als $j \neq k$. In het asymmetrische geval kunnen we iets dergelijks toepassen. In §7 zullen we verder op deze methode in gaan.

Een bezwaar tegen de vorige methode is dat de rekentijd en de vereiste geheugenruimte ongeveer een factor 3 groter wordt, wanneer het aantal steden 1 toeneemt. Deze factor is veel kleiner voor de volgende methode, terwijl de programmeerbaarheid daarbij even goed blijft. Deze z.g. "branch and bound method" kan ook op verschillende andere problemen toegepast worden, [25]:

De verzameling van alle toegelaten oplossingen wordt telkens in kleinere stukken verdeeld en er wordt een ondergrens berekend voor de kosten van de oplossingen in elke deelverzameling. Na deze verdeling worden die deelverzamelingen met een ondergrens die niet kleiner is dan de kosten van een bekende, goede oplossing voor verder onderzoek uitgesloten. Het vertakken gaat door tot er nog maar één oplossing over is.

In 1958 had Eastman deze techniek reeds toegepast op het handelsreizigersprobleem, [13], [14]. Hij stelde voor het corresponderende toekenningsprobleem op te lossen, waarvoor wel een goede oplossingsprocedure bestaat. Is de gevonden permutatie niet cyclisch, dan is de gevonden waarde een ondergrens voor de kortste route afstand. We kiezen dan de kleinste cykel in de permutatie. Een van de k schakels daarin mag in de uiteindelijke oplossing niet voortkomen. Dus worden k nieuwe toekenningsproblemen gevormd, voor elk van de k schakels één, waarin verboden wordt deze schakel in de ronde op te nemen. De optimale oplossing is van tenminste één van de nieuwe problemen een toegelaten oplossing. De nieuwe problemen worden daarna opgelost, nieuwe ondergrenzen berekend en de vertakking wordt voortgezet aan de tak met de kleinste ondergrens. Daarna wordt het proces voortgezet.

In 1963 verscheen een artikel van Little en de dames Murty, Sweeney en Karel, [26], waarin de "branch and bound" techniek ook toegepast wordt. Hun criterium voor het splitsen in deelverzamelingen is echter essentieel anders. Sweeney wijdde haar proefschrift aan het programmeren van deze methode, [40]. De rekentijden blijken veel korter dan van andere methoden die optimaliteit garanderen (zie §10).

Hun uitgangsgedachte is de volgende: zoek in de te splitsen deelverzameling die schakel die het meeste voordeel geeft en splits nu in ronden die dit stuk weg wel en die het niet bevatten. De eerste deelverzameling is nu aanzienlijk kleiner dan de deelverzameling waarvan we uitgaan, de tweede heeft in de meeste gevallen een betrekkelijk hoge ondergrens. In §8 zullen we op deze methode verder in gaan.

# HOOFDSTUK 2. OPLOSSINGSMETHODEN

We zullen in dit hoofdstuk een aantal oplossingsmethoden nader bestuderen. Om te laten zien hoe de methode werkt zullen we steeds proberen er een probleem mee op te lossen. Opdat de vergelijking van de methoden zo goed mogelijk zal zijn, zullen we steeds hetzelfde probleem nemen. Het is een 10-steden probleem ontleend aan Barachet, [3]. De symmetrische afstandsmatrix staat in tabel I. De situatie kan ook vrij goed in de twee dimensionale Euclidische ruimte weergegeven worden (figuur 1), hoewel, waarschijnlijk door afronding, niet helemaal aan de driehoeksongelijkheid is voldaan (b.v. $C_{23} + C_{36} < C_{26}$).

| Stad | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 28 | 57 | 72 | 81 | 85 | 80 | 113 | 89 | 80 |
| 2 | 28 | 0 | 28 | 45 | 54 | 57 | 63 | 85 | 63 | 63 |
| 3 | 57 | 28 | 0 | 20 | 30 | 28 | 57 | 57 | 40 | 57 |
| 4 | 72 | 45 | 20 | 0 | 10 | 20 | 72 | 45 | 20 | 45 |
| 5 | 81 | 54 | 30 | 10 | 0 | 22 | 81 | 41 | 10 | 41 |
| 6 | 85 | 57 | 28 | 20 | 22 | 0 | 63 | 28 | 28 | 63 |
| 7 | 80 | 63 | 57 | 72 | 81 | 63 | 0 | 80 | 89 | 113 |
| 8 | 113 | 85 | 57 | 45 | 41 | 28 | 80 | 0 | 40 | 80 |
| 9 | 89 | 63 | 40 | 20 | 10 | 28 | 89 | 40 | 0 | 40 |
| 0 | 80 | 63 | 57 | 45 | 41 | 63 | 113 | 80 | 40 | 0 |

TABEL I

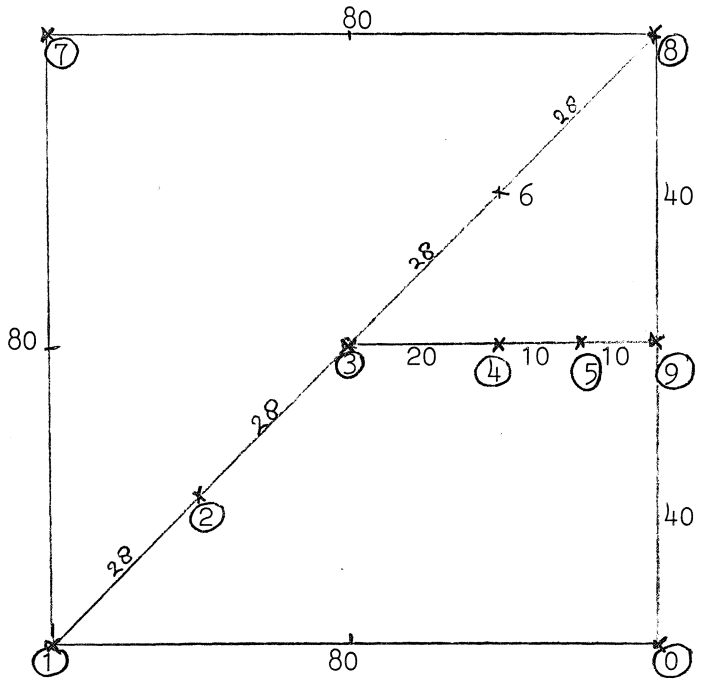
Afstandsmatrix van probleem van Barachet

## §3. Een meetkundige methode

Litt. L.L. Barachet, Graphic Solution of the Traveling Salesman Problem [3],
R.H. Gonzalez, Solution of the Traveling Salesman Problem by D.P. [18].

De methode van Barachet werd pas in 1957 gepubliceerd. Toch hoort ze thuis in de periode dat computers nog niet het belangrijkste middel waren bij het vinden van de beste route. Essentieel is n.l. dat we een figuur hebben, waarin de lengte van een schakel evenredig is met de afstand tussen de beeldpunten. De methode is b.v. niet toepasbaar in het asymmetrische geval of het geval waarin niet aan de driehoeksongelijkheid voldaan is (denk aan ordeningsproblemen).

Barachet begint met een cyclische permutatie van de steden 1,2,...,n, die b.v. op het oog gekozen is, en leidt dan een rij steeds kortere ronden af, eerst een waarvoor elk van de n groepjes van drie opeenvolgende schakels minimaal is, vervolgens een waarvoor elk van de n verbindingen bestaande uit vier schakels minimaal is, enz., totdat men zo de goede eindoplossing verkrijgt.




figuur 1. Probleem van Barachet

Om hierbij te bepalen of twee bepaalde steden verbonden moeten worden om de optimale route te krijgen, past Barachet de volgende regels toe:

1) De optimale ronde doorsnijdt zichzelf niet.
2) Als de binnenhoeken gevormd door drie opeenvolgende schakels stomp zijn, dan vormen deze drie schakels een kortste verbinding voor de vier betreffende steden.
3) Als de steden de hoekpunten zijn van een convex polygoon, vormen de zijden van dat polygoon de beste ronde.
4) Een route bestaande uit n-k opeenvolgende schakels en aansluitend een minimale verbinding van k segmenten heeft een lengte die tenminste gelijk is aan de lengte van die laatste verbinding.
5) In het geval 1 t/m 4 geen uitsluitsel geeft berekenen we de kortste verbinding.

Numeriek voorbeeld:

De beginroute is (0123456789) met totale afstand 411. De eerste stappen geven we aan in tabel II, waarbij > kortere verbinding, < langere verbinding betekent. De startplaats en aankomstplaats van een verbinding worden als vast beschouwd:

| | | | |
|---|---|---|---|
| p = 3: | 5678 | < | 5768 |
| | 6789 | < | 6879 |
| p = 4: | 23456 | < | 25436 |
| | 45678 | < | 47658 |
| | " | < | 47568 |
| | 56789 | < | 57869 |
| | 67890 | < | 69870 |
| p = 5: | 123456 | < | 154236 |
| | 234567 | < | 245637 |
| | " | < | 254637 |
| | " | < | 265437 |
| | 345678 | > | 374568 |
| | nieuwe route: (0123745689) | | |
| p = 3: | 2374 | > | 2734 |
| | nieuwe route: (0127345689) | | |
| p = 3: | 4568 | < | 4658 |
| | | enz. | |

TABEL II. Begin (regel 5)

p = 3: We beschouwen de 10 groepjes van 3 opeenvolgende schakels van de beginronde: 1234, 2345, 3456 en 4567 zijn minimaal volgens regel 3; 5678 kan alleen vergeleken worden met 5768, maar is korter, dus er is geen correctie nodig; Analoog is 6789 korter dan 6879. 7890, 8901, 9012 en 0123 zijn minimaal volgens regel 3.

p = 4: 12345: regel 3; 23456: 2 kan slechts met 5 verbonden worden (regel 1) dus vergelijk 23456 met 25436 etc.

.....

De eerste moeilijkheid treedt op bij het geval p = 5:

p = 5: .... 345678: 3 kan alleen met 7 verbonden worden (regel 1), dus keuze tussen 374568, 376458 en 374658, de laatste wordt geelimineerd (regel 2 op 7465), 374568 is korter, dus nieuwe route: (01237456890).

We starten voor deze route weer bij p = 3:

p = 3: 0123: minimaal volgens regel 3 enz.

In tabel III geven we de verkortingen aan. Elke keer moet men dan weer bij p = 3 beginnen. De uiteindelijke ronde wordt (0543217689) met een totale lengte 378.

| | | |
|---|---|---|
| p = 5: | 345678 → | 374568 |
| 3: | 2374 → | 2734 |
| 4: | 45689 → | 46859 |
| 6: | 2734685 → | 2378645 |
| 8: | 378645901 → | 345098671 |

TABEL III. Verbeteringen van de route

Met totaal 175 optellingen van maximaal 9 termen elk wordt het eindresultaat bereikt. Het is duidelijk dat dit onvoldoende de te verrichten hoeveelheid werk weergeeft, daar het toepassen van de regels 1,2,3 en 4 hier niet in verwerkt is.

## §4. Toepassing van de Simplexmethode

Litt. Dantzig, Fulkerson en Johnson: Solution of a Large Scale Traveling Salesman Problem, [10].
Dantzig, Fulkerson en Johnson: On a Linear-Programming, Combinatorial Approach to the Traveling Salesman Problem, [11].

Deze methode berust op de theorie van de lineaire programmering. Eerst wordt een beginroute gekozen. Vervolgens leiden we een aantal lineaire vergelijkingen en ongelijkheden af, waaraan door alle toegelaten ronden voldaan is. Met de Simplexmethode berekenen we een nieuwe basisoplossing. Is deze oplossing ook een toegelaten oplossing van het handelsreizigersprobleem, dan is het blijkbaar de beste route en in dat geval zijn we klaar. In andere gevallen voegen we nieuwe lineaire eisen toe, zodat de gevonden basisoplossing uitgesloten wordt, maar alle toegelaten ronden ook voldoen aan de nieuwe eisen. Vervolgens wordt de Simplexmethode weer toegepast, enz. Tijdens dit proces is het soms handig een schattingsprocedure te gebruiken om een aantal permutaties met te grote lengte voor verder onderzoek uit te sluiten. Wanneer er nog weinig mogelijkheden resten, kunnen we de optimale oplossing berekenen op grond van combinatorische argumenten of door de lengte van alle nog niet uitgesloten ronden te berekenen en te vergelijken (computer).

Stel nu dat $C = (C_{ij})_{i,j=1}^{n}$ een symmetrische afstandsmatrix is en dat K een bepaalde route langs de steden 1,2,...,n is. Als stad $j$ in K direct verbonden is met stad $i$ stellen we $x_{ij} = 1$, anders $x_{ij} = 0$. De kosten van de route is dan $\sum_{\substack{i,j=1 \\ i>j}}^{n} x_{ij}C_{ij}$. Verder geldt $x_{ii} = 0$, $i = 1,2,\ldots,n$. Elke stad is met twee andere steden direct verbonden, dus $\sum_{i=1}^{n} x_{ij} = 2$, $j = 1,2,\ldots,n$.

Het uitgangspunt van Dantzig, Fulkerson en Johnson is nu het volgende probleem:

$$\text{minimaliseer} \quad z = \sum_{\substack{i,j=1 \\ i>j}}^{n} x_{ij}C_{ij}$$

onder (1) $\sum_{i=1}^{n} x_{ij} = 2, \quad j = 1,2,\ldots,n,$

(2) $x_{ij} \geq 0, \; i,j = 1,2,\ldots,n.$

Dat dit probleem te weinig voorwaarden bevat blijkt direct uit het feit dat elke oplossing van het toekenningsprobleem ook aan de voorwaarden voldoet. De eenvoudigste typen toe te voegen eisen zijn $x_{i_0 j_0} \leq 1$ en $x_{i_1 i_2} + x_{i_2 i_3} + \ldots + x_{i_r i_1} \leq r - 1$, de eerste om $x_{i_0 j_0}$ niet willekeurig groot te laten worden, de tweede om de cykel $(i_1 i_2 \ldots i_r)$ in de eindroute uit te sluiten. Er zijn echter ook meer gecompliceerde eisen die soms toegevoegd moeten worden, b.v. voor basisoplossingen met niet gehele getallen als uitkomst. In ons te behandelen numeriek voorbeeld zal echter het stellen van bovengrenzen aan de variabelen ons in staat stellen met behulp van de schattingsprocedure en combinatorische argumenten de beste oplossing te vinden.

Numeriek voorbeeld:

Op verzoek hebben Dantzig, Fulkerson en Johnson een tweede artikel, [11], aan hun methode gewijd. Hierin hebben ze uitgewerkt, hoe ze de optimale oplossing van het probleem van Barachet bepalen:

De uitgangsroute is (1234567890). Allereerst willen we een aantal voorwaarden krijgen waarvoor deze route een basisoplossing is. Daarom eisen we $x_{12} \leq 1$ (blok op de schakel in fig. 2). Verder vervangen D., F. en J. om hun methode gemakkelijker uit te kunnen leggen $x_{12}$ met waarde 1 als basisvariabele door $x_{68}$ met waarde 0.

De volgende stap is nu z.g. potentialen $p_i$ (prijzen) zo te bepalen dat $p_i + p_j = C_{ij}$, voor alle paren (i,j) die corresponderen met basisvariabelen $x_{ij}$. Oplossen van een aantal eenvoudige lineaire vergelijkingen geeft:

$p_1 = 57\frac{1}{2}$, $p_2 = 1\frac{1}{2}$, $p_3 = 26\frac{1}{2}$, $p_4 = -6\frac{1}{2}$, $p_5 = 16\frac{1}{2}$, $p_6 = 5\frac{1}{2}$, $p_7 = 57\frac{1}{2}$, $p_8 = 22\frac{1}{2}$, $p_9 = 17\frac{1}{2}$ en $p_0 = 22\frac{1}{2}$.

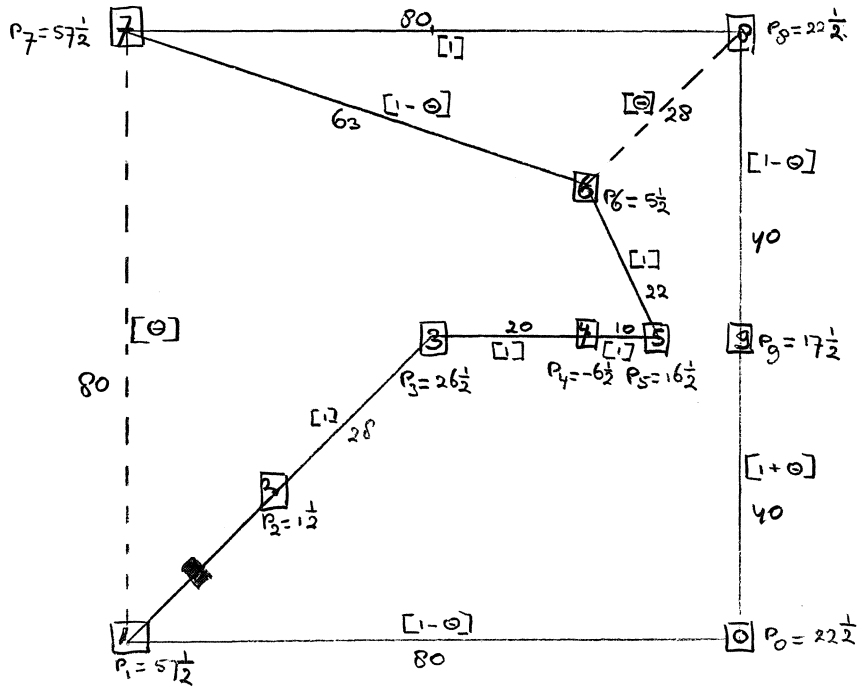



fig. 2

Hieruit berekenen we $d_{ij}$:

$$d_{ij} = \begin{cases} p_i + p_j - C_{ij} & \text{als } x_{ij} = 0 \\ C_{ij} - p_i - p_j & \text{als } x_{ij} > 0. \end{cases}$$

We zoeken nu de waarden van i en j waarvoor $d_{ij}$ maximaal is, stel M. Is $M \leq 0$, dan is de gevonden oplossing optimaal en zijn we klaar. Als $M > 0$ is, kiezen we volgens de Simplexmethode het corresponderende element $x_{ij}$ als nieuw basiselement. In ons geval vinden we i = 1, j = 7, M = 35. Stellen we $x_{ij} = \theta$, dan volgt uit de voorwaarden (1) en (2), (zie fig. 2) dat $x_{09} = 1 + \theta \geq 1$. Daaruit volgt $\theta = 0$ en we laten $x_{09}$ weg uit de basis.

We beginnen nu de nieuwe waarden voor de potentialen te berekenen om bovenstaand procédé te herhalen. In fig. 3 zien we de resultaten van deze stap: i = 0, j = 5, M = 33.

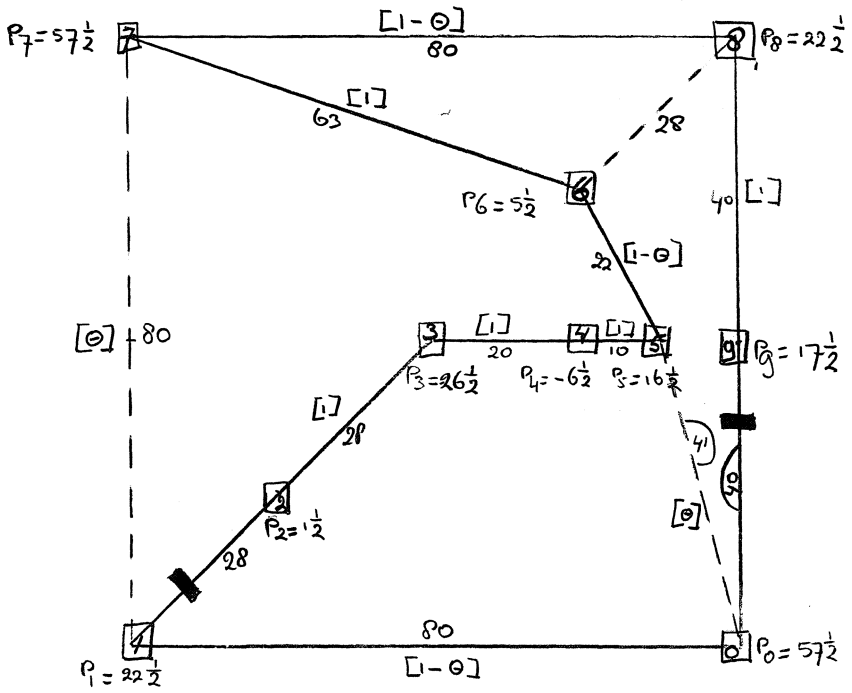



fig. 3

Deze keer kunnen we θ de waarde 1 geven zonder met (1) of (2) in conflict te komen. Zo krijgen we een nieuwe route (1234509867) welke 33 korter is dan onze beginroute. $x_{01}$ (of $x_{56}$ of $x_{78}$) wordt nu uit de basis verwijderd.

Na nog drie van deze stappen is de som van de positieve $d_{ij}$ gelijk aan 7. Er zijn dan nog slechts acht schakels die niet tot de basis behoren en een waarde $d_{ij} \leq 6$ hebben. Men kan gemakkelijk nagaan dat de andere schakels "te duur" zijn en verder niet hoeven worden beschouwd. Uit het feit dat twee andere schakels die niet in de basis zitten en positieve $x_{ij}$ waarde hebben, een potentiaal $\geq 7$ hebben, volgt dat deze zeker in de eindoplossing moeten optreden. Hierdoor vallen er volgens (1) vier schakels af. Men bewijst nu gemakkelijk dat de andere schakels ook geen verbetering geven en dat (01234509867) de optimale ronde is.

## 5. Oplossen met inversies

Litt. G.A. Croes, A method for Solving Traveling Salesman Problems, [7].

In 1958 publiceerde Croes een methode om het handelsreizigersprobleem op te lossen, die uit drie stappen bestaat:

1) het bepalen van een beginoplossing.
2) het verbeteren van deze oplossing met behulp van inversies.
3) een verder onderzoek om de beste oplossing te bepalen.

Zijn methode is alleen bruikbaar voor symmetrische matrices.

Tijdens de eerste stap ordenen we de elementen van matrix C naar grootte. Voor het probleem van Barachet krijgen we dus de schakels 45(10), 59(10), 34(20), 49(20), 46(20), 56(22) enz. (Tussen haakjes staat de lengte van de schakels). Tijdens de aftelling bewaren we een element alleen, als er een route bestaat die naast dat element ook alle elementen bevat die we al eerder bewaard hebben. 49 laten we dus weg, omdat er geen ronde bestaat die de schakels 45,59 en 49 bevat. 46 laten we weg omdat 4 al verbonden is met 5 en 3. Om analoge reden sluiten we 56 uit. Tellen we verder af dan krijgen we:
45(10), 59(10), 34(20), 12(28), 23(28), 68(28), 69(28), 17(80), 80(80) en 70(113). We vinden zo de ronde (1234596807) met lengte 425.

In de tweede stap zoeken we naar inversies die deze oplossing verbeteren. Een inversie is een verandering van een route $(i_1 i_2 \cdots i_{j-1} i_j i_{j+1} \cdots i_{k-1} i_k i_{k+1} \cdots i_n)$ in een route $(i_1 i_2 \cdots i_{j-1} i_k i_{k-1} \cdots i_{j+1} i_j i_{k+1} \cdots i_n)$. Opdat deze ronde korter is moet vanwege de symmetrie gelden

$$C_{j-1,k} + C_{j,k+1} < C_{j-1,j} + C_{k,k+1}. \tag{3}$$

| Stad | 1 | 2 | 3 | 4 | 5 | 9 | 6 | 8 | 0 | 7 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | - | 28 | 57 | 72 | 81 | 89 | 85 | 113 | 80 | 80 |
| 2 | | - | 28 | 45 | 54 | 63 | 57 | 85 | 63 | 63 |
| 3 | | | - | 20 | 30 | 40 | 28 | 57 | 57 | 57 |
| 4 | | | | - | 10 | 22 | 20 | 45 | 45 | 72 |
| 5 | | | | | - | 10 | 22 | 41 | 41 | 81 |
| 9 | | | | | | - | 28 | 40 | 40 | 89 |
| 6 | | | | | | | - | 28 | 63 | 63 |
| 8 | | | | | | | | - | 80 | 80 |
| 0 | | | | | | | | | - | 113 |
| 7 | | | | | | | | | | - |

TABEL IV

In tabel IV staat het probleem van Barachet met onze beginoplossing in de <u>bovendiagonaal</u>, d.w.z. de elementen $A_{i,i+1}$, $i = 1,2,\ldots,n-1$, terwijl het laatste element van de oplossing in de rechterbovenhoek staat. De inversies die een verbetering geven hebben in (3) de volgende waarden voor j en k: (5,8), (9,8), (6,8), (3,0), (4,0) en (6,0). De verbeteringen zijn resp. 4,9,5,21,4 en 38. We voeren daarom de laatste inversie uit. Er is geen tweede inversie die we tegelijkertijd uitvoeren, daar voor de andere inversies geldt $j' \leq j \leq k' \leq k$ of $j \leq j' \leq k \leq k'$. De nieuwe lengte wordt 425 - 38 = 387.

| Stad | 1 | 2 | 3 | 4 | 5 | 9 | 0 | 8 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | - | 28 | 57 | 72 | 81 | 89 | 80 | 113 | 85 | 80 |
| 2 | | - | 28 | 45 | 54 | 63 | 63 | 85 | 57 | 63 |
| 3 | | | - | 20 | 30 | 40 | 57 | 57 | 28 | 57 |
| 4 | | | | - | 10 | 22 | 45 | 45 | 20 | 72 |
| 5 | | | | | - | 10 | 41 | 41 | 22 | 81 |
| 9 | | | | | | - | 40 | 40 | 28 | 89 |
| 0 | | | | | | | - | 80 | 63 | 113 |
| 8 | | | | | | | | - | 28 | 80 |
| 6 | | | | | | | | | - | 63 |

TABEL V

In tabel V staat de situatie na deze inversie, de nieuwe ronde (1234590867) in de bovendiagonaal. Inversies die nog een verbetering geven zijn (5,0) en (9,0) met verbetering resp. 5 en 9. Uitvoeren van de tweede inversie levert de optimale oplossing (1234509867) met lengte 378. Inverteren kan soms echter ook vrij slechte resultaten geven: de oplossing (1234567890) is door inversies niet meer te verbeteren, hoewel de lengte 33 groter is dan de afstand van de beste oplossing.

In de derde stap wordt getracht van zo veel mogelijk schakels te bewijzen dat deze te lang zijn om in de eindoplossing voor te komen. Hiertoe brengen we de route die tot dan toe het kortste is weer in de hoofddiagonaal. Bij deze beschouwingen spelen de elementen van de inverse ronde een bijzondere rol. Vanwege de onoverzichtelijkheid van deze stap zullen we er hier niet verder op in gaan.

## §6. Het opbouwen van een goede oplossing.

Litt. R.L. Karg en G.L. Thompson, An Heuristic Approach to Solving the Traveling Salesman Problem, [23].

Karg en Thompson stellen in hun artikel voor een goede oplossing te berekenen met een zeer eenvoudige procedure. De verkregen oplossing hoeft weliswaar niet optimaal te zijn, maar de methode is zelfs voor grote problemen (50 à 60 steden) bruikbaar.

We zullen het principe weer aan het probleem van Barachet duidelijk trachten te maken. We tellen de steden af in de volgorde 1,2,3,4,5,6,7,8,9,0, maar elke andere aftelling is ook goed. Vanwege de symmetrie is elke ronde langs de steden 1,2 en 3 even lang. Voor de steden 1,2,3 en 4 zijn er drie verschillende mogelijkheden: (1234), (1423) en (1243). Er geldt, als z de kosten aangeeft:

$$z(1234) - z(123) = C_{34} + C_{41} - C_{31},$$

$$z(1234) - z(123) = C_{24} + C_{43} - C_{23},$$

$$z(1423) - z(123) = C_{14} + C_{42} - C_{12}$$

We berekenen het minimum van de rechterleden en vinden zo de ronde langs deze steden met de kleinste lengte. Uitgaande van deze ronde

passen we stad 5 zo voordelig mogelijk in. Hiervoor moeten we het minimum van vier termen berekenen. Zetten we dit proces voort en brengen we de resultaten in een schema, dan krijgen we:

| | | |
|---|---|---|
| | (123) | lengte 113., route (123) |
| k=4: | 142,243,[341] | d.w.z. 4 inpassen tussen 3 en 1 |
| Δz: | 89, 37,[35] | nieuwe lengte 148, route (1234) |
| k=5: | 152,253,354,[451] | |
| Δz: | 107, 56, 20,[19] | z = 167, route (12345). |
| k=6: | 162,263,364,465,[561] | |
| Δz: | 114, 57, 28, 32,[26] | z = 193, route (123456) |
| k=7: | 172,273,374,475,576,[671] | |
| Δz: | 115, 92,109,143,139,[58] | z = 251, route (1234567) |
| k=8: | 182,283,384,485,586,[687], 781 | |
| Δz: | 170,114, 82, 76, 47,[45], 113 | z = 296, route (12345687) |
| k=9: | 192,293,394,495,[596],698,897,791 | |
| Δz: | 124, 75, 40, 20,[16], 49, 49, 98 | z = 312, route (123459687) |
| k=10: | 102,203,304,405,[509],906,608,807,701 | |
| Δz: | 115, 92, 82, 76,[71], 75,115,113,113 | z = 383, route (1234509687) |

In dit schema geeft k de volgende stad aan die ingepast moet worden en geeft Δz de waarde van het corresponderende rechterlid aan. We vinden dus een lengte 383, 5 langer dan de beste route. Waren we uitgegaan van de aftelling 1,6,2,7,3,8,4,9,5,0 dan hadden we ook dit resultaat verkregen.

Het aantal operaties bij deze methode is evenredig met $n^2$, terwijl voor de meeste andere methoden de rekentijd exponentieel toeneemt, bij toename van het aantal steden. Omdat de resultaten nogal sterk afhangen van de volgorde van aftellen van de steden, stellen zij voor het resultaat te berekenen voor een aantal aftellingen. Zonder daar verder op in

te gaan vermelden we dat Karg en Thompson hun theorie hebben uitgebreid met een methode om een groot probleem te splitsen, deze kleinere problemen eerst afzonderlijk te onderzoeken en daarna de resultaten te combineren. In §10 worden de rekentijden vergeleken met die, nodig bij andere methoden.

## 7. Toepassing van de dynamische programmering

Litt. R.H. Gonzalez: Solution of the Traveling Salesman Problem by Dynamic Programming on the Hypercube, [18].
M. Held en R.M. Karp: A Dynamic Programming Approach to Sequencing Problems, [20].

We geven een formulering van het handelsreizigersprobleem als dynamisch programmeringsprobleem:
Zij S een deelverzameling van $\{2,3,\ldots,n\}$, $\ell \in S$ en $z(S,\ell)$ de minimale kosten om startend in stad 1 en alle steden van S bezoekend in stad $\ell$ te eindigen. Er geldt:

(4) $\#\, S = 1$: $\quad z(S,\ell) = C_{1\ell} \qquad \ell = 2,3,\ldots,n,$

(5) $\#\, S > 1$: $\quad z(S,\ell) = \min_{m \in S\setminus\ell} (z(S\setminus\ell,m) + C_{m\ell}).$

($\#\, S$ betekent: het aantal elementen van S). Zij $S_0 = \{2,3,\ldots,n\}$, dan is $z(S_0 \cup 1,1)$ de kortste route afstand en geldt

(6) $$z(S_0 \cup 1,1) = \min_{\ell \in S_0} (z(S_0,\ell) + C_{\ell 1}).$$

De juistheid van deze formules laat zich gemakkelijk nagaan. Uit (4), (5) en (6) volgt direct:
Een permutatie $(1,i_2,\ldots,i_n)$ is dan en slechts dan optimaal als

(7) $$z(S_0 \cup 1,1) = z(S_0,i_n) + C_{i_n 1}$$

en voor $2 \leq p \leq n-1$

(8) $$z(\{i_2,i_3,\ldots,i_p,i_{p+1}\},i_{p+1}) = z(\{i_2,i_3,\ldots,i_p\},i_p) + C_{i_p i_{p+1}}.$$

De berekening verloopt nu in twee fasen:
Eerst worden de $z(S,\ell)$ recursief uitgerekend m.b.v. (4) en (5) en tenslotte de minimale kosten met (6). In de tweede fase worden met (7) en (8) achtereenvolgens $i_n, i_{n-1}, i_{n-2}, \ldots, i_1$ bepaald. Zoals reeds in §2 is vermeld, krijgen we een verkorting door halverwege de resultaten te combineren.

Held en Karp hebben in [20] ook een methode gegeven om een goede oplossing te geven bij problemen van meer dan dertien steden. Deze bestaat uit verwisselingen van globale en locale verbeteringen van een beginoplossing. Bij de globale verbetering wordt bovenstaande methode toegepast op 13 zoveel mogelijk even grote groepjes steden, bij de locale wordt nagegaan of verwisseling van twee opeenvolgende steden een verbetering geeft. Hun resultaten zijn in §10 vermeld.

De methode voor problemen van minder dan veertien steden gaat niet uit van een beginoplossing, dit in tegenstelling tot de meeste andere hier genoemde methoden. Structureel kunnen we precies het aantal benodigde operaties en het aantal benodigde geheugenplaatsen als functie van n berekenen. In de dissertatie van Gonzalez zijn uitvoerige berekeningen opgenomen en een kritische beschouwing van de berekeningen van Bellman, [18], blz. 9-11. Bij ongeveer 15 steden blijken zowel de rekentijd als de benodigde geheugencapaciteit te groot te worden om het probleem met de computer op te lossen.

Numeriek voorbeeld:
In het volgende betekent $\overline{i_2 i_3 \ldots i_{k-1} i_k}$ de kortste verbinding van stad 1 naar stad $i_k$ langs de plaatsen $i_2, i_3, \ldots, i_{k-1}$ in een willekeurige volgorde. Een schema van het begin van het oplossen van het probleem van Barachet ziet er dan als volgt uit:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| k=1 | S: | 2, | 3, | 4, | 5, | 6, | 7, | 8, | 9, | 0 | |
| kosten | z: | 28 | 57 | 72 | 81 | 85 | 80 | 113 | 89 | 80 | |
| k=2 | S: | $\overline{23}$, | $\overline{24}$, | $\overline{25}$, | $\overline{26}$, | $\overline{27}$, | $\overline{28}$, | $\overline{29}$, | $\overline{20}$, | $\overline{32}$, | $\overline{34}$, | $\overline{35}$,... |
| kosten | z: | 56, | 73, | 82, | 85, | 91, | 113, | 91, | 91, | 85, | 77, | 87,... |
| k=3 | S: | $\overline{234}$, | $\overline{235}$, | $\overline{236}$, | $\overline{237}$, | $\overline{238}$, | $\overline{239}$, | $\overline{230}$, | $\overline{243}$, | $\overline{245}$, | $\overline{246}$,... | |
| kosten | z: | 76, | 86, | 84, | 113, | 113, | 96, | 113, | 93, | 83, | 93,... | |
| | | ~~130~~, | ~~139~~, | ~~142~~, | ~~148~~, | ~~170~~, | ~~148~~, | ~~148~~, | ~~145~~, | ~~147~~, | ~~145~~,... | |

(de eerste waarde van z geldt voor de routes 1234, 1235,..., de tweede voor de routes 1324, 1325,...; beide worden berekend uit de resultaten van het geval k=2, waarna we de kleinste afstand kiezen).

k=4 S: $\overline{2345}$, $\overline{2346}$, ...

kosten z: 86 96

(de kosten van b.v. $\overline{2345}$ worden verkregen door uit k=3 af te leiden $z(\overline{234}) + z(45) = 86$, $z(\overline{243}) + z(35) = 123$, $z(\overline{342}) + z(25) = 228$ en daarna de laagste kosten te kiezen. Het is duidelijk dat voortzetting van dit proces tot de optimale oplossing leidt).

## §8. Een branch and bound methode.

Litt. Little, Murty, Sweeney en Karel, An Algorithm for the Traveling Salesman Problem, [26].
Lawler en Wood, Branch and Bound Methods, a Survey, [25].

Een voorbeeld van het bepalen van een ondergrens voor een groep oplossingen hebben we al in de vorige paragraaf gezien. $z(\overline{234}) = 76$ impliceert dat alle routes die beginnen met 1234 of 1324 tenminste 76 eenheden lang zijn; $z(\overline{2345}) = 86$ houdt in dat een route waarvan 1 de eerste en 5 de vijfde plaats is en waarbij voor stad 5 de steden 2,3 en 4 gepasseerd worden tenminste 86 lang is. Uiteindelijk vinden we dat elke route tenminste 378 eenheden meet. Daar de route (1234509867) deze lengte heeft is deze route blijkbaar optimaal.

Bij een branch and bound methode worden de collecties ronden waarvoor we een ondergrens gevonden hebben telkens vertakt in kleinere verzamelingen ronden. Verreweg de beste resultaten tot nog toe zijn verkregen door een methode die in 1963 door Little, Murty, Sweeney en Karel gepubliceerd werd. Zij streven er naar een groep oplossingen te splitsen in een relatief kleine groep met een lage ondergrens en een grote groep met een relatief hoge ondergrens.

Maar reeds in 1958 waren twee methoden ontwikkeld die min of meer

van deze techniek gebruik maken. Rossman, Twery en Stone passen in een ongepubliceerd artikel ([36], zie ook [37]) ideeën toe, die ze combinatorische programmering noemen. Hiermee losten ze een 13-steden probleem in 8 mandagen op (rekentijd voor hetzelfde probleem door Little e.a. $3\frac{1}{2}$ uur). Eastman geeft in [13], [14] (vgl. [25]) een methode die gebruik maakt van de correspondentie met het toekenningsprobleem. Voor het laatste type problemen bestaan n.l. zeer snelle oplossingsmethoden ([1], blz. 149-150), en elke oplossing van het handelsreizigersprobleem is ook een oplossing voor het corresponderende toekenningsprobleem (hierbij wordt $C_{ii} = \infty$ gesteld). Door dit laatste op te lossen vindt Eastman een ondergrens voor het eerste probleem. Is de gevonden oplossing cyclisch dan zijn we klaar, in het andere geval nemen we de kleinste cykel en vormen we een nieuw probleem voor elke schakel in die cykel, waarbij deze schakel in de oplossing verboden wordt. Hierbij is de oplossing van het handelsreizigersprobleem ook een oplossing van tenminste een van de nieuwe problemen. Het procédé wordt voortgezet totdat we een cyclische oplossing vinden met een lengte die niet groter is dan de ondergrenzen van de andere groepen oplossingen. Op deze methode gaan we niet verder in. Alvorens op de methode van Little e.a. in te gaan vermelden we nog dat S.N.N. Pandit in [29] belooft een nieuwe branch and bound methode in [30] te publiceren.

Stel dat $(i_1 i_2 \ldots i_n)$ de oplossing is van het handelsreizigersprobleem en dat de kortste lengte z is. Als we nu alle elementen van een bepaalde rij of kolom van de kostenmatrix C met een vaste waarde x verminderen, dan zal $(i_1 i_2 \ldots i_n)$ ook de beste oplossing zijn van het nieuwe probleem, terwijl de lengte van de nieuwe ronde z-x is. Dit volgt direct uit het feit dat een willekeurige ronde uit elke rij (kolom) precies een element bevat.

Door nu eerst van alle rijen het kleinste element uit die rij af te trekken en dit vervolgens ook voor alle kolommen te doen, krijgen we een matrix met niet-negatieve elementen waarbij in elke rij en in elke kolom een nul voorkomt. Zo'n matrix noemen we een <u>gereduceerde matrix</u>. Als s de som is van de reductiegetallen, dan is $(i_1 i_2 \ldots i_n)$ ook de optimale oplossing van het probleem van de gereduceerde matrix

en is de lengte van deze oplossing z -s. Reduceren we de matrix van het probleem van Barachet, dan krijgen we de matrix in tabel VI. Uit s=324 volgt dat 324 een ondergrens is voor de afstand voor alle ronden (vgl. lengte optimale oplossing is 378).

| kolom / rij | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | reductie: | G: |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ∞ | 0 | 29 | 44 | 53 | 57 | 17 | 77 | 61 | 22 | 28 | 17 |
| 2 | 0 | ∞ | 0 | 17 | 26 | 29 | 0 | 49 | 35 | 5 | 28 | 0 |
| 3 | 37 | 8 | ∞ | 0 | 10 | 8 | 2 | 29 | 20 | 7 | 20 | 2 |
| 4 | 62 | 35 | 10 | ∞ | 0 | 10 | 27 | 27 | 10 | 5 | 10 | 5 |
| 5 | 71 | 44 | 20 | 0 | ∞ | 12 | 36 | 23 | 0 | 1 | 10 | 0 |
| 6 | 65 | 37 | 8 | 0 | 2 | ∞ | 8 | 0 | 8 | 13 | 20 | 0 |
| 7 | 23 | 6 | 0 | 15 | 24 | 6 | ∞ | 15 | 32 | 26 | 57 | 6 |
| 8 | 85 | 57 | 29 | 17 | 13 | 0 | 17 | ∞ | 12 | 22 | 28 | 12 |
| 9 | 79 | 53 | 30 | 10 | 0 | 18 | 44 | 22 | ∞ | 0 | 10 | 0 |
| 0 | 40 | 23 | 17 | 5 | 1 | 23 | 38 | 32 | 0 | ∞ | 40 | 1 |
| reductie: | _ | _ | _ | _ | _ | _ | 35 | 8 | _ | 30 | | |
| H: | 23 | 6 | 0 | 0 | 0 | 6 | 2 | 15 | 0 | 1 | s=324 | |

TABEL VI

In de volgende stap berekenen we van elke rij resp. kolom het op een na kleinste element. Deze noteren we resp. in de kolom onder G en in de rij volgend op H. We vragen ons nu af welke schakel het slechtst gemist kan worden. Wanneer we niet van 1 naar 2 gaan, dan moeten we van 1 naar een andere plaats, wat minstens G[1] = 17 kost. Verder moeten we naar 2 vanuit een andere stad dan 1, wat tenminste H[2] = 6 kost. Totaal dus 23. In het algemeen krijgen we dus als $C_{ij} = 0$ dat G[i] + H[j] de prijs u is die we moeten betalen als we niet van i naar j gaan. Maximalizeren we deze sommen dan zien we dat het maximum u = 23 is, bereikt voor 1 → 2 en 2 → 1. We splitsen nu alle routes in twee groepen: de ene groep is de groep die alle ronden bevat die 1 → 2 niet als schakel bezitten. Vanwege de symmetrie van de matrix sluiten we ook 2 → 1 uit; we zouden

anders de inverse ronde niet uitsluiten en nog zeker een optimale oplossing in deze grote groep hebben. Uitsluiting van 1 → 2 en 2 → 1 kost tweemaal 23 en we vinden dus dat een route die in het oorspronkelijke probleem niet van 1 naar 2 of vice versa gaat tenminste 324+46 = 370 lang is. De andere groep is de groep ronden waarin de schakel 1 → 2 wel optreedt. We hoeven dan niet meer van 1 naar een andere plaats en kunnen dus de eerste rij schrappen. Zo kunnen we ook de tweede kolom schrappen. Verder stellen we $C_{21} = \infty$ daar we de cykel 1 → 2 → 1 willen uitsluiten. Reduceren we de 9 × 9 matrix dan vinden we de matrix in tabel VII. We hebben hierbij voor 23 gereduceerd, zodat we als ondergrens voor routes die 1 → 2 bevatten 324+23 = 347 krijgen.

| Kolom / rij | 1 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | reductie | G |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | ∞ | 0 | 17 | 26 | 29 | 0 | 49 | 35 | 5 | - | 0 |
| 3 | 14 | ∞ | 0 | 10 | 8 | 2 | 29 | 20 | 7 | - | 2 |
| 4 | 39 | 10 | ∞ | 0 | 10 | 27 | 27 | 10 | 5 | - | 5 |
| 5 | 48 | 20 | 0 | ∞ | 12 | 36 | 23 | 0 | 1 | - | 0 |
| 6 | 42 | 8 | 0 | 2 | ∞ | 8 | 0 | 8 | 13 | - | 0 |
| 7 | 0 | 0 | 15 | 24 | 6 | ∞ | 15 | 32 | 26 | - | 0 |
| 8 | 52 | 29 | 17 | 13 | 0 | 17 | ∞ | 12 | 22 | - | 12 |
| 9 | 56 | 30 | 10 | 0 | 18 | 44 | 22 | ∞ | 0 | - | 0 |
| 0 | 17 | 17 | 5 | 1 | 23 | 38 | 32 | 0 | ∞ | - | 1 |
| reductie: | 23 | - | - | - | - | - | - | - | - | | |
| H: | 14 | 0 | 0 | 0 | 6 | 2 | 15 | 0 | 1 | s = | 347 |

TABEL VII

We maken een geheugenboom (fig. 4) waarin we de splitsing en van elke groep de ondergrens noteren.

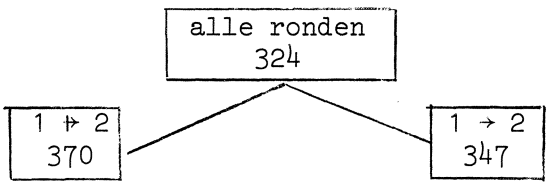



fig. 4

We gaan met de tak met de laagste ondergrens verder; in ons geval is dat dus de rechtertak. We berekenen weer het op een na kleinste element in elke rij en kolom van tabel VII. De schakel die nu het slechtst gemist kan worden is $8 \rightarrow 6$. Dat kost 18. Voor de rechtertak stellen we $C_{68} = \infty$ om $6 \rightarrow 8 \rightarrow 6$ te vermijden.

Het proces voortzettend krijgen we als volgende splitsing $7 \not\rightarrow 1$ of $7 \rightarrow 1$. In de rechtertak moeten we nu niet $C_{17} = \infty$ stellen. Dit element is n.l. al uitgesloten: in deze tak hebben we de schakels $1 \rightarrow 2$ en $7 \rightarrow 1$ zodat we $2 \rightarrow 7$ moeten uitsluiten. We stellen dus $C_{27} = \infty$ voor deze tak.

Op dit moment hebben we de situatie in fig. 5. We moeten nu verder gaan met de tak verkregen door $1 \rightarrow 2$, $8 \not\rightarrow 6$ met ondergrens 365. Daartoe gaan we terug naar tabel VII en stellen $C_{86} = \infty$. Reduceren van de matrix brengt de ondergrens op 365. We gaan van hieruit

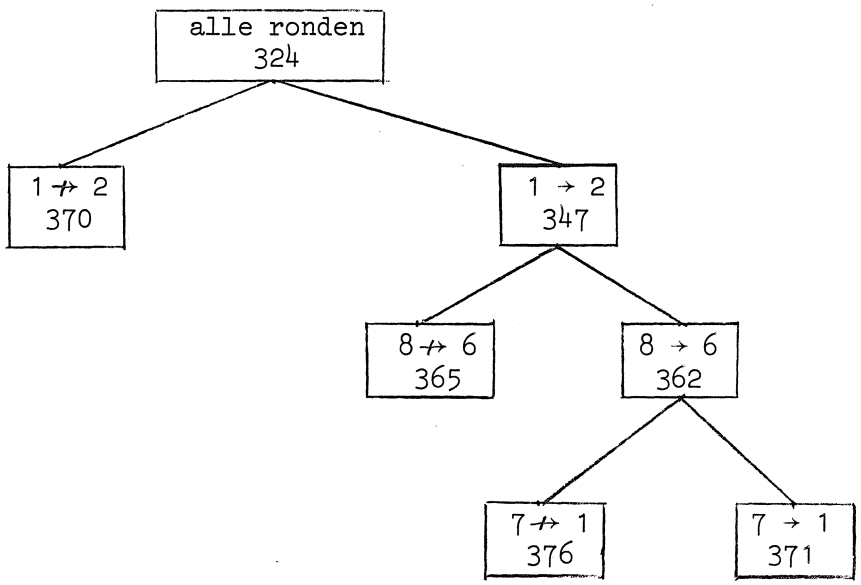
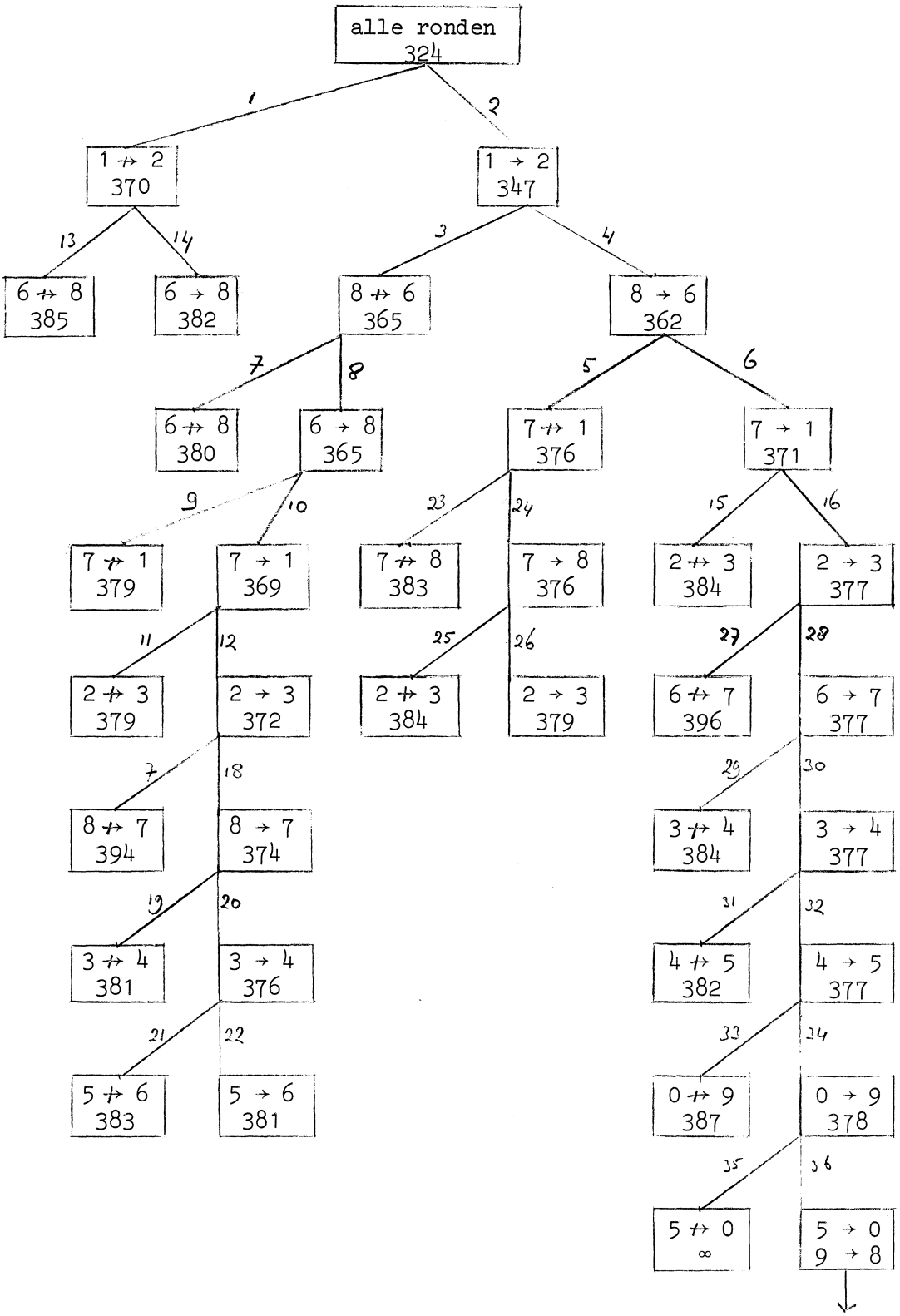



fig. 5

door met vertakken totdat alle takken een ondergrens > 370 hebben. We gaan daarna verder met tak $1 \not\rightarrow 2$. Daartoe stellen we in tabel VI $C_{12} = \infty$. Maar vanwege de symmetrie sluiten we om de reeds genoemde reden ook $2 \not\rightarrow 1$ uit. Reductie geeft nu een matrix met ondergrens 370. Daarna keren we weer naar de rechtertak terug ($1 \rightarrow 2$, $8 \rightarrow 6$, $7 \rightarrow 1$). We zetten dit proces voort totdat we een $2 \times 2$-matrix overhouden.




optimale route (1234509867)

afstand 378.

fig. 6

Zo vinden we als optimale route (1234509867) met lengte 378.
In fig. 6 staat de boom na afloop van de berekeningen, langs de takken hebben we de volgorde weergegeven waarin deze takken ontsproten zijn. Een ALGOL programma dat deze methode in gewijzigde vorm toepast zal in §9 besproken worden.

Er zijn veel variaties op het thema mogelijk. We kunnen daarbij twee gevallen onderscheiden: in het eerste geval beginnen we steeds met de tak met de laagste ondergrens, in het tweede geval maken we eerst de rechtertak af om een goede oplossing te hebben, beginnen aan een nieuwe tak en maken deze naar rechts af tenzij de ondergrens voor die tijd al groter dan of gelijk aan de lengte van de eerste oplossing is, enz.

In het eerste geval kunnen we alle situaties onthouden die we nog nodig kunnen hebben, hetgeen bij grote problemen hoge eisen kan stellen aan de geheugencapaciteit, of telkens de tabel bij een tak opnieuw afleiden uit de beginsituatie.

In het tweede geval maken we gebruik van het feit dat vertakken naar rechts veel minder werk is dan overschakelen op een andere tak. Hierbij lopen we het risico een tak helemaal te berekenen terwijl de ondergrenzen veel te groot worden en de optimale oplossing in een andere tak zit. Verder is er natuurlijk ook een mengstrategie voor beide gevallen te bedenken.

We kunnen een variatie op het tweede geval geven die een zeer aanzienlijke reductie geeft in de benodigde geheugenruimte. We werken weer eerst de rechtertak af, bewaren bij elke vertakking de situatie en vinden tenslotte een oplossing met afstand 378. We gaan nu van beneden naar boven terug langs de linkertakken tot we een ondergrens < 378 vinden (in ons geval aan tak 5) en we gaan nu hiermee verder. We hoeven alle gepasseerde linkertakken niet meer te onthouden. Tak 5 is snel uitgeput, ook de andere linkervertakkingen (eerst alles na tak 3, dan tak 1) hoeven niet meer onthouden te worden daar de ondergrens te hoog is, maar anders zou de vrijgekomen geheugenruimte daarvoor gebruikt kunnen worden. Bij deze methode, voorgesteld door Little e.a. ([26], blz. 984) is om de matrices te onthouden een geheugenruimte $\sum_{k=3}^{n} k^2 = \frac{1}{6}(n^3+3n^2+2n) - 5$ ver-

eist. In §9 geven we een variatie, die ten koste van wat meer rekenwerk voor het onthouden van de matrices een geheugenruimte $n^2$ eist.

Er zijn gevallen waarin het zeer lang kan duren voordat we de beste oplossing gevonden hebben. Dat is b.v. het geval wanneer in alle kolommen en rijen meerdere nullen voorkomen. Daardoor zullen alle elementen van G en H nul worden en de keuze van de schakel volledig willekeurig worden. Deze situatie kan optreden bij een ordeningsprobleem (vgl. §1), waarbij een aantal groepen producten zijn, zodat het niets kost om de machine van één product op een ander product uit dezelfde groep om te schakelen. Bij de methode van §9 zullen we aan de hand van een voorbeeld zien dat we in dat geval iets minder last hebben, maar het zou pas ideaal zijn als we een methode hadden die van deze nullen gebruik maakt.

We willen deze paragraaf besluiten met een aantal nog niet genoemde mogelijkheden van deze oplossingsmethode:

1) We hebben een methode aangegeven die werkt voor asymmetrische matrices en met een kleine wijziging (ondergrens eerste linkertak) voor symmetrische matrices. We zouden het procédé zo kunnen veranderen dat we het paar (i,j) als niet geordend beschouwen. Dit zou een besparing geven in rekentijd en benodigde geheugenruimte.
2) Bij het berekenen van het op een na kleinste element van elke rij en kolom kunnen we ophouden met het onderzoeken van een rij of kolom zodra we twee elementen daarin gevonden hebben die kleiner zijn dan of gelijk aan het maximum van de op een na kleinste elementen van eerder onderzochte rijen of kolommen.
3) Stel dat gevraagd wordt naar een oplossing die niet langer is dan b.v. 103% van de optimale oplossing. Bij het probleem van Barachet zouden we dan na de rechtsaftakking alleen de takken met onderwaarde $< \frac{100}{103} \cdot 378 = 367$ nog hoeven te onderzoeken. Analoog zouden we een flinke reductie krijgen als gegeven was dat onze oplossing hoogstens 10 langer mag zijn dan de optimale oplossing.
4) Door cykels niet te verbieden lossen we het corresponderende toekenningsprobleem op.

# HOOFDSTUK 3: PROGRAMMA EN RESULTATEN

## §9: Een ALGOL programma.

Het programma dat we in deze paragraaf behandelen is gebaseerd op de branch and bound methode van Little e.a., die we in §8 behandeld hebben. De theoretische waarde van het programma is waarschijnlijk groter dan de practische. Enerzijds zou een aanzienlijke hoeveelheid onderzoek nodig zijn om te bepalen welke variant gemiddeld het minste rekenwerk zal geven en zou het gebruik van machinetaal eveneens de rekentijd verminderen, anderzijds is de door het programma vereiste geheugenruimte in zekere zin minimaal.

Zoals we al gezien hebben blijft de beste route van een matrix optimaal als we een vast getal bij alle elementen van een rij of kolom optellen. Twee matrices die in elkaar overgevoerd kunnen worden door een aantal keren een getal bij de elementen van een rij of kolom op te tellen zullen we equivalent noemen.

We nemen aan dat de afstanden een grootte orde hebben, die veel kleiner is dan $_{10}9$. Tijdens het programma zal de afstandenmatrix C steeds modulo $_{10}9$ equivalent zijn met de oorspronkelijke matrix. We zullen $_{10}9$ bij een element optellen als het volgens de theorie $\infty$ moet worden. Zo zullen we steeds voldoende hebben aan de n × n-matrix en hoeven we verder geen ruimte te reserveren om afstanden te onthouden.

We beginnen met het aftakken naar rechts om een goede beginoplossing te krijgen. Bij reductie verminderen we echter alle elementen van de rij of kolom met het reductiegetal. z is steeds de som van alle reductiewaarden, zodat de lengte van een route in de oorspronkelijke matrix z meer is dan de lengte van dezelfde route in de nieuwe matrix. Verder onthouden we de elementen waar we $_{10}9$ bij opgeteld hebben om cykels te vermijden in de (geschiedenis) arrays E1 en E2, terwijl de ondergrenzen van de linkertakken in het array F bewaard worden. Om naar rechts af te takken maken we gebruik van de procedures REROW en REDUROW om rijen te reduceren, RECOL om een kolom te reduceren en het op een na kleinste element in die kolom te bepalen, ZEROW om het op een na kleinste element in een rij te bepalen, STRIGHT om de schakel te bepalen welke

we in de route opnemen en om langs de rechtertak naar het volgende vertakkingspunt te gaan.

Wanneer we aan het einde van de rechtsaftakking gekomen zijn geven de schakels A[i] → B[i] de gevonden oplossing aan, welke oplossing we door procedure NEW z1 in array K in routevolgorde opslaan tot we een betere oplossing hebben.

We bepalen de laagste linkertak waarvan we nog moeten aftakken omdat dit een betere oplossing zou kunnen geven en daarna trekken we in procedure LEBRAN $_{10}9$ af van de elementen die na dat vertakkingspunt in de rechtertak verboden zijn en tellen we $_{10}9$ op bij de schakel die in de linkertak uitgesloten wordt. Merk op dat we nu na reductie een andere ondergrens kunnen vinden. Is ook deze ondergrens te laag dan vertakken we weer naar rechts totdat we een betere oplossing of een te hoge ondergrens vinden. We zetten het proces voort totdat we van een oplossing de optimaliteit bewezen hebben.

Deze methode lijkt een voordeel te hebben in gevallen waarbij in elke rij en kolom meerdere nullen voorkomen. De keuze van een schakel is dan willekeurig en omdat de oude methode steeds van deelmatrices van de oorspronkelijke matrix uitgaat is de kans groot dat eerst een aanzienlijk aantal schakels verboden moet worden, voordat we de optimaliteit van een oplossing kunnen bewijzen. Bij het voortzetten van het proces met de equivalente matrix zullen de nullen waarschijnlijk sneller geen rol meer spelen.

Vergelijk b.v. de afstandenmatrices in tabel VIII en tabel IX. Zoals we al in §2 vermeld hebben kan in sommige gevallen door steden in groepjes te beschouwen een aanzienlijke reductie in het rekenwerk verkregen worden. Onderscheid n.l. in tabel VIII de steden 1,2,3,4 en 5 als groep 1 en 6,7,8,9 en 10 als groep 2. Het kost dan niets om binnen zo'n groep van stad tot stad te gaan.

| rij \ Kolom | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ∞ | 0 | 0 | 0 | 0 | 12 | 15 | 8 | 13 | 6 |
| 2 | 0 | ∞ | 0 | 0 | 0 | 23 | 27 | 18 | 12 | 15 |
| 3 | 0 | 0 | ∞ | 0 | 0 | 10 | 9 | 27 | 8 | 9 |
| 4 | 0 | 0 | 0 | ∞ | 0 | 13 | 15 | 8 | 20 | 17 |
| 5 | 0 | 0 | 0 | 0 | ∞ | 17 | 12 | 14 | 8 | 10 |
| 6 | 12 | 23 | 10 | 13 | 17 | ∞ | 0 | 0 | 0 | 0 |
| 7 | 15 | 7 | 9 | 15 | 12 | 0 | ∞ | 0 | 0 | 0 |
| 8 | 8 | 18 | 27 | 8 | 14 | 0 | 0 | ∞ | 0 | 0 |
| 9 | 13 | 12 | 8 | 20 | 8 | 0 | 0 | 0 | ∞ | 0 |
| 0 | 16 | 15 | 9 | 17 | 10 | 0 | 0 | 0 | 0 | ∞ |

TABEL VIII

Stellen we een afstandstabel op voor de groepen dan krijgen we

| | groep 1 | groep 2 |
|---|---|---|
| groep 1 | ∞ | 6 |
| groep 2 | 7 | ∞ |

We zullen minstens een keer van groep 1 naar groep 2 en eenmaal terug moeten keren. Dus is de minimale lengte 13, b.v. verkregen door de route (1,10,6,9,8,7,2,3,5,4).

Deze route is blijkbaar ook de optimale ronde van het probleem bij matrix IX, maar er lijkt geen algemene methode te bestaan die deze oplossing op een snelle wijze als de beste aanwijst.

| rij \ Kolom | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ∞ | 0 | 8 | 0 | 0 | 12 | 15 | 8 | 13 | 6 |
| 2 | 0 | ∞ | 0 | 4 | 0 | 23 | 27 | 18 | 12 | 15 |
| 3 | 9 | 0 | ∞ | 11 | 0 | 10 | 9 | 27 | 8 | 9 |
| 4 | 0 | 6 | 0 | ∞ | 0 | 13 | 15 | 8 | 20 | 17 |
| 5 | 0 | 0 | 16 | 0 | ∞ | 17 | 12 | 14 | 8 | 10 |
| 6 | 12 | 23 | 10 | 13 | 17 | ∞ | 0 | 0 | 0 | 0 |
| 7 | 15 | 7 | 9 | 15 | 12 | 0 | ∞ | 0 | 0 | 0 |
| 8 | 8 | 18 | 27 | 8 | 14 | 0 | 0 | ∞ | 12 | 0 |
| 9 | 13 | 12 | 8 | 20 | 8 | 0 | 0 | 0 | ∞ | 0 |
| 0 | 16 | 15 | 9 | 17 | 10 | 0 | 0 | 3 | 4 | ∞ |

TABEL IX

Tenslotte merken we op dat, wanneer we eenmaal een goede oplossing gevonden hebben, we soms het rekenwerk kunnen bekorten. Stel n.l. dat we een goede oplossing met lengte z1 gevonden hebben, en dat de ondergrens van de matrix bij een zeker vertakkingspunt z is, z < z1. Wanneer we nu bij het berekenen van het op een na kleinste element van een rij of kolom vinden dat dit element groter is dan z1 - z, dan weten we, dat we dit element in de oplossing moeten opnemen en dat de linkertak niet onderzocht hoeft te worden. Dus mogen we zonder naar een betere schakel te zoeken naar het volgende vertakkingspunt langs de rechtertak gaan.

begin comment dit programma tracht het handelsreizigersprobleem op te lossen m.b.v. de branch and bound methode volgens Little e.a.
Eerst wordt het aantal op te lossen problemen ingelezen, vervolgens te beginnen met het grootste probleem het aantal plaatsen n bij het eerste probleem, de matrix C die de afstanden weergeeft, gegeven in de volgorde C[1,1], C[1,2], ..., C[1,n], C[2,1], C[2,2], ..., C[2,n], ..., C[n,1], C[n,2], ..., C[n,n] en tenslotte de Boolean

symm die true wordt dan en slechts dan als C symmetrisch is. In dat geval wordt een 1 op het invoerbandje geponst, in het andere geval een 0. Bij meer problemen wordt dan weer het aantal plaatsen van het tweede probleem ingelezen, de bijbehorende matrix C enz. Nadat een probleem opgelost is wordt de minimale routelengte geprint, gevolgd door een rangschikking van de plaatsen waarbij deze afstand bereikt wordt en tenslotte de tijd gebruikt voor de berekening;

```
integer a,b,c,d,e,f,g,h,i,j,k,l,m,n,p,q,r,s;
real    t,u,v,w,x,y,z,vv,t1,z1,t2;
Boolean symm;
    s:=read; n:=read;
begin  integer array A,B,K[1:n], E1,E2,[1:n×n];
       real array  C[1:n,1:n], F,G,H[1:n];

       procedure WSL (e,f); integer e,f; begin d:=e; e:=f; f:=d  end;
       procedure PLUS (v); real v; v:=v+10^9;
       procedure MIN (v);  real v; v:=v-10^9;

       procedure REDUROW (m); value m; integer m;
       begin   x:=C[m,B[1]];
               for j:=k step -1 until 2 do if x > C[m,B[j]] then x:=C[m,B[j]];
               if x=0 then go to L1 else for j:=n step -1 until 1 do
               C[m,j]:=C[m,j]-x;
       L1:     z:=z+x
       end;

       procedure REROW (m); value m; integer m;
       if G[i]>0 then
       begin   x:=G[i]; z:=z+x;
               for j:=n step -1 until 1 do C[m,j]:= C[m,j]-x
       end;

       procedure RECOL (m); value m; integer m;
       begin   x:= C[A[1],m]; y:=10^10;
               for i:=k step -1 until 2 do
               begin v:= C[A[1],m];
```

```
                if v<x then begin y:=x; x:=v end
                else if v < y then begin y:=v;
                if y=0 then go to L2 end
        end;
        if x > 0 then
        begin for i:=n step -1 until 1 do C[i,m]:=C[i,m]-x;
                y:=y-x; z:=z+x
        end;
L2:     if w < y then begin w:=y; c:=j end; H[j]:=y
end;

procedure ZEROW (m); value m; integer m;
begin   x:=C[m,B[1]]; y:=10^10;
        for j:=k step -1 until 2 do
        begin v:=C[m,B[j]];
                if v < x then begin y:=x, x:=v end else
                if v < y then begin y:=v; if y=0 then go to L3 end
        end;
L3:     G[i]:=y
end;

procedure LEBRAN
begin for j:=p step -1 until 1 do if E1[j] < 1000 then
        MIN (C[E1[j], E2[j]]) else if E1[j]=1000 + i then
        begin E1[j]:=a:=A[i]; E2[j]:=b:=B[i];
                PLUS (C[a,b]);
                p:=j; k:=l:=i; vv:=0;
                for r:=k step -1 until 1 do REDUROW (A[r]);
                go to L8
        end
end;
```

```
procedure STRIGHT (r); value r; integer r;
begin u:=-1;
        for i:= r step -1 until 1 do if G[i] < 0 then
        for j:=r step -1 until 1 do if C[A[i], B[j]] = 0 then
        begin t:=G[i] + H[j]; if t > u then
              begin u:=t; g:=i; h:=j end;
              j:=0
        end;
        if u < w then
        begin u:=w; h:=c; b:=B[c];
              for i:=r step -1 until 1 do if C[A[i],b] = 0 then
              begin g:=i; i:=0 end
        end;
        F[r]:= z+u; p:=p+2; E1[p-1]:= E2[p-1]:=1000+r;
        G[g]:= G[r]; WSL(A[g],A[r]); a:=d;
        WSL (B[h], B[r]); b:=d;
        for i:=p-2 step -1 until 2 do if E2[i]=b then
        begin E1[p]:= E1[i]; if E1[i-1] > 1000 then go to L4
        end;
        E1[p]:=b;
L4:     for i:=p-2 step -1 until 2 do if E1[i]=a then
        begin E2[p]:= E2[i]; if E2[i-1] > 1000 then go to L5
        end;
        E2[p]:= a;
L5:     PLUS C(E1[p], E2[p])
end;

procedure NEWz1;
begin   if C[A[1], B[1]] > 10^8 then WSL (A[1], A[2]);
        K[1]:=A[n]; f:=K[2]:=B[n]; e:=3;
L6:     for i:=1 step 1 until n do if A[i]=f then
        begin f:=K[e]:=B[i]; e:=e+1;
              if e ≤ n then go to L6 else go to L7
        end;
L7:     z1:=z
end;
```

```
L0:     t1:=time; k:=l:=n; z:=vv:=0;z1:=10 10; p:=0;
        for i:=1 step 1 until n do
        begin for j:=1 step 1 until n do C[i,j]:= read;
              A[i]:=B[i]:= i;    PLUS(C[i,i])
        end;
        m:=read; symn:=m=1;
        for i:=n step -1 until 1 do REDUROW (i);
L8:     for k:=l step -1 until 3 do
        begin w:=-1; if vv:=10 9 then REDUROW (E1[p]);
              for j:=k step -1 until 1 do RECOL (B[j]);
              if z1 < z then begin q:=k; F(q):=z; go to L9 end;
              for i:=k step -1 until 1 do ZEROW (A[i]);
              STRIGHT (k); vv:=C[E1[p], E2[p]];
              if H[h]=0 then for i:=k-1 step -1 until 1 do
              if C[A[i], b]=0 then REROW(A[i])
        end;
        q:=k:=2;. if vv:=10 9 then REDUROW (E1[p]);
        if z < z1 then NEW z1;
L9:     for i:=q+1 step 1 until n-1 do if F[i] < z1 then LEBRAN;
        if F[n] < z1 then
        begin if symn then PLUS (C[B[n], A[n]]); symn:=false;
              i:=n; LEBRAN
        end;
        t2:=time;
        PRINTTEXT ({afstand}); PRINT(z1); NLCR;
        PRINTTEXT ({route}); for i:=1 step 1 until n do ABSFIXT
                                                    (3,0,K[i]);
        NLCR;
        PRINTTEXT ({rekentijd}); PRINT (t2-t1); NLCR; NLCR;
        s:=s-1; if s > 0 then begin n:=read; go to L0 end
    end
end
```

## §10. Vergelijking van rekentijden

Verschillende auteurs hebben om de snelheid van hun methode aan te geven rekentijden gegeven, die nodig waren om het probleem tot een goed einde te brengen. In de loop van de tijd zijn zo een aantal problemen ontstaan die als testprobleem geschikt zijn de methoden met elkaar te vergelijken. Bij het gebruik van een computer maakt het type computer deze vergelijking natuurlijk weer moeilijker. De problemen zijn:

I 10 steden probleem van Barachet, [3], symmetrisch.

II 49 steden probleem van Dantzig, Fulkerson en Johnson, [10], symmetrisch.

III 20 steden probleem van Croes, [7], symmetrisch.

IV 48 velden paarden rit op 6 × 8 "schaak"bord.

V 64 velden paarden rit op 8 × 8 schaakbord

VI 10 problemen van 10 steden, Robacker, [35], symmetrisch.

VII 25 steden probleem van Held en Karp, [20], symmetrisch.

VIII 48 steden probleem van Held en Karp, [20], symmetrisch.

De resultaten waren:

| | Auteurs/ Methode | Manier van berekenen | Benodigde tijd | Berekende opl. | Optimaliteit bewezen | Optimale oplossing |
|---|---|---|---|---|---|---|
| I | Gonzalez [18] | IBM 1620 | 8 min. 48 sec. | 378 | ja | 378 |
| I | Dacey [8] | Hand | ? | 3,2% te groot | neen | 378 |
| I | Programma §9 | X8 | 1.7 sec | 378 | ja | 378 |
| II | Croes [7] | Hand | 70 uur | 699 | ja | 699 |
| II | Dacey [8] | Hand | 90 min. | 727 | neen | 699 |
| II | Held en Karp [20] | IBM 7090 | $2 \leq x \leq 15$ min. | 699 | neen | 699 |
| II | Karg en Thompson [23] | Bendix G-20 | 4.5 min. | 699 | neen | 699 |

| | Auteurs/ Methode | Manier van berekenen | Benodigde tijd | Berekende opl. | Optimaliteit bewezen | Optimale oplossing |
|---|---|---|---|---|---|---|
| III | Dacey [8] | Hand | ? | 265 | neen | 246 |
| III | Held en Karp [20] | IBM 7090 | $2 \leq x \leq 15$ min. | 246 | neen | 246 |
| III | Little e.a. [26] | IBM 7090 | .126 min. | 246 | ja | 246 |
| III | Programma §9 | X8 | .45 min. | 246 | ja | 246 |
| IV | Held en Karp [20] | IBM 7090 | $2 \leq x \leq 15$ min. | 52 | neen | 48 |
| V | Little e.a. [26] | IBM 7090 | .178 min. | 64 | ja | 64 |
| VI | Croes [7] | Hand | Gemiddeld 25 min. | | ja | |
| VI | Dacey [8] | Hand | gemiddeld 5 min. | gem. 4,8% te groot | neen | |
| VI | Robacker, [35] met methode Dantzig, Fulkerson en Johnson | Hand | gemiddeld 3 uur | | ja | |
| VII | Held en Karp [20] | IBM 7090 | $2 \leq x \leq 15$ min. | 1711 | neen | 1711 |
| VII | Little e.a. [26] | IBM 7090 | 4.7 min. | 1711 | ja | 1711 |
| VIII | Held en Karp [20] | IBM 7090 | $2 \leq x \leq 15$ min. | 11.566 | neen | ? |
| VIII | Little e.a. [26] | IBM 7090 | $\geq$ 50 min. | ? | neen | ? |

Er zijn ook enkele auteurs, die een aantal even grote problemen onderzochten en een gemiddelde tijd opgaven:

| | | | | |
|---|---|---|---|---|
| Gonzalez [18], | 7 steden, | IBM 1620, | 29 sec., | optimaliteit bewezen |
| Gonzalez [18], | 10 steden, | IBM 1620, | 8 min. 48 sec., | " " |
| Little e.a. [26], | 10 steden, | Hand, | 1 uur, | " " |
| Little e.a. [26], | 10 steden, | IBM 7090, | 0.012 min., | " " |
| Held en Karp [20], | 13 steden, | IBM 7090, | 17 sec., | " " |
| Little e.a. [26], | 20 steden, | IBM 7090, | 0.084 min., | " " |
| " " | 30 steden, | IBM 7090, | 0.957 min., | " " |
| " " | 40 steden, | IBM 7090, | 8.37 min., | " " |

De waarden die Little, Murty, Sweeney en Karel geven zijn de gemiddelden over problemen met asymmetrische matrices met aselect getrokken getallen als afstanden.

Literatuur

1. Ackoff, R.L., Progress in Operations Research, Vol. I, Wiley, New York, 1961.
2. Ball, W.W.R., Mathematical Recreations and Essays, New York, 1939.
3. Barachet, L.L., Graphic Solution of the Traveling Salesman Problem, O.R., 5 (1957), 841-845.
4. Bellman, R.E., Combinatorial Processes and Dynamic Programming, Proceedings of the Tenth Symposium in Appl. Math. of the Am. Math. Soc. 1960.
5. ———, Dynamic Programming Treatment of the Traveling Salesman Problem, J. Assoc. Comput. Mach., 9 (1962), pp. 61-63.
6. Bock, F., An Algorithm for solving the Traveling Salesman Problem and Related Network Optimization Problems, Armour Research Foundation, Technical Center, Chicago 16, Illinois. Abstract in O.R., 6 (1958), pp. 897.
7. Croes, G.A., A Method for Solving Traveling Salesman Problems, O.R., 6 (1958), pp. 791-814.
8. Dacey, M.F., Selection of an Initial Solution of the Traveling Salesman Problem, Mimeographed for private distribution, available from the Department of Geography, University of Washington, 1959.
   Abstract in O.R., 8 (1960), pp. 133-134.
9. Dantzig, G.B., Discrete variable extremum problems, O.R., 5 (1957), 266-276.
10. Dantzig, G.B., Fulkerson, D.R.,and Johnson, S.M., Solution of a Large-Scale Traveling Salesman Problem, O.R., 2 (1954), pp. 393-410.
11. Dantzig, G.B., Fulkerson, D.R., and Johnson, S.M., On a Linear-Programming, Combinatorial Approach to the Traveling-Salesman Problem, O.R., 7 (1959), pp. 58-66.
12. Doig, A. and Land, A.H., An Automatic Method of Solving Discrete Programming Problems, Econometrica, 28 (1960), pp. 497-520.
13. Eastman, W.L., Linear Programming with Pattern Constraints, Ph.D. dissertation, Harvard University, July 1958; also in augmented form:Report No. BL-20 The Computation Laboratory, Harvard University, July 1958.

14. Eastman, W.L., A Solution to the Traveling Salesman Problem, Econometrica, 27 (1959), pp. 282.

15. Flood, M.M., The Traveling Salesman Problem, O.R., 4 (1956), 61-75.

16. Gilmore, P.C. and Gomory, R.E., A Solvable Case of the Traveling-Salesman Problem, Proc. of the National Acad. of Sciences, 51 (1964), pp. 178-181.

17. Gilmore, P.C. and Gomory, R.E., Sequencing a one state-variable machine. A Solvable Case of the Traveling Salesman Problem, O.R., 12 (1964), 655-679.

18. Gonzalez Zubieta, R.H., Solution of the Traveling Salesman Problem by Dynamic Programming on the Hypercube, Interim Technical Report No. 18, O.R. Center, M.I.T., 1962.

19. Hardgrave, W.W. and Newhauser, G., On the Relation between the Traveling Salesman Problem and the Longest Path Problem, O.R., 10 (1962), pp. 647-657.

20. Held, M. and Karp, R.M., A Dynamic Programming Approach to Sequencing Problems, J. Soc. Indust. and Appl. Math., 10 (1962), pp. 196-210.

21. Heller, I., The Traveling-Salesman Problem, Part I: Basic Facts, The George Washington U. Logistics Research Project, 1954.

22. ———, On the Traveling Salesman Problem, Proc. of the Second Symposium in Linear Programming, Vol. 2, pp. 643-665.

23. Karg, R.L. and Thompson, G.L., An Heuristic Approach to Solving the Traveling Salesman Problem, Management Science, 10 (1964), pp. 225-248.

24. Kuhn, H.W., The Traveling Salesman Problem, Proc. Sixth Symposium in Appl. Math. of the Am. Math. Soc.

25. Lawler, E.L. and Wood, D.E., Branch and Bound Methods, a Survey, O.R., 14 (1966), pp. 699-719.

26. Little, J.D.C., Murty, K.G., Sweeney, D.W. and Karel, C., An Algorithm for the Traveling Salesman Problem, O.R., 11 (1963), pp. 972-989.

27. Miller, C.E., Tucker, A.W. and Zemlin, R.A., Integer Programming Formulation of the Traveling Salesman Problem, J. Assoc. Comp. Mach., 7 (1960), pp. 326-329.

28. Mudrov, V.I., A Method of Solution of the Traveling Salesman Problem by means of Integer Linear Programming, Zhurnal Vychislennoĭ Matem. i. Matem. Fiziki, USSR, 3 (1963), pp. 1137-1139, vertaald door R.F. Rinehart, Ref. Zhur. Matem. (1964).

29. Pandit, S.N.N., Some Observations on the Longest Path Problem, O.R., 12 (1964), pp. 361-364.

30. ———, Search for Optimal Nonlooped Paths (to appear).

31. Piehler, J., A contribution to sequencing problems, Unternehmungsforschung, 4 (1960), pp. 138-142.

32. Pollack, M. and Weibenson, W., Solution of the Shortest Route Problem, O.R., 8 (1960), pp. 224-230.

33. Riley III, W., A New Approach to the Traveling Salesman Problem, B. O.R.S.A., O.R., 7 (1959), Suppl. 1:B-11.

34. ———, Micro-analysis applied to the Traveling Salesman Problem, B.O.R.S.A., O.R., 8 (1960), Suppl. 1:B-12.

35. Robacker, J.T., Some experiments of the Traveling Salesman Problem, Rand Report, RM 1521, 1955.

36. Rossman, M.J. and Twery, J., Combinatorial Programming, presented at the 6th Annual ORSA meeting, May 1958, mimeographed.

37. Rossman, M.J., Twery, R.J. and Stone, F.D., A Solution to the Traveling Salesman Problem by Combinatorial Programming, mimeographed, abstract in O.R., 6 (1958), p. 897.

38. Rothkopf, M., The Traveling Salesman Problem: On the Reduction of certain Large Problems to Smaller Ones, O.R., 14 (1966), pp. 532-533.

39. Saksena, J.P. and Kumar, S., The Routing Problem with K specified nodes, O.R., 14 (1966), pp. 909-913.

40. Sweeney, D.W., The Exploration of a New Algorithm for Solving the Traveling Salesman Problem, M.S. Thesis, M.I.T., 1963.

41. Tucker, A.W., On Directed Graphs and Integer Programs, Princeton I.B.M. Math. Research Proj., Techn. Report 1960.

42. Votaw, D.F. and Orden, A., The Personnel Assignment Problem, Symposium on Linear Inequalities and Programming, project SCOOP, Hq. U.S. Air Force, 1952, pp. 155-163.

43. Wisler, C.E., An Approach to the Sequencing of Range Operators, U.S. Gouvernment, Research Report AD-402612.