STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

AFDELING MATHEMATISCHE STATISTIEK

S 415 (SB5)

# Een oplossingsmethode voor het lokomotiefdienstregelingprobleem

door

B. Nederkoorn

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

september 1969

*Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam, The Netherlands.*

*The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications; it is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O) and the Central Organization for Applied Scientific Research in the Netherlands (T.N.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.*

# Een oplossingsmethode voor het lokomotiefdienstregelingprobleem

door

B. Nederkoorn

Inhoud

## 1. Inleiding

De Nederlandse Spoorwegen hebben een dienstregeling voor treinen bestemd voor goederenvervoer. Elke trein bestaat uit een zeker aantal wagons en wordt voortgetrokken door een lokomotief. Door de dienstregeling zijn de treinen vastgelegd, de lokomotieven echter nog niet. Teneinde deze treinen '24 uur rond' te rijden zijn een minimum aantal lokomotieven nodig. Uiteraard is het onwaarschijnlijk dat alle ritten precies op elkaar aansluiten, zodat de lokomotieven vaak, nadat ze in stad A een rit hebben beëindigd, naar een andere stad B moeten rijden teneinde de volgende rit te beginnen. De onrendabele kilometers die de lokomotief moet afleggen om van de ene rit naar de andere te gaan worden 'leegloop-km' genoemd. Bij gebruik van een zeker aantal lokomotieven zal een minimum aantal leegloopkm moeten worden gereden.
Een essentiële bemoeilijking bij het vinden van een minimum aantal lokomotieven en het met dit aantal corresponderende minimum aantal leegloop-km is het feit dat elke lokomotief eens in de zoveel dagen een onderhoudsbeurt nodig heeft.
In dit rapport zal een branch-and-bound algorithme aangegeven worden dat voor elk aantal lokomotieven en elk aantal onderhoudsbeurten het minimum aantal te rijden leegloopkm berekent, alsmede de routes die moeten worden gekozen om dit minimum aantal te bereiken. Op deze wijze wordt een systeem verkregen dat het maximum aantal te besparen leegloop-km aangeeft bij toevoeging van telkens één of meerdere lokomotieven. Verder is opgenomen een programmatekst van dit algorithme in ALGOL 60 alsmede een summiere toelichting daarop. Daar het algorithme slechts voor een dienstregeling met een betrekkelijk gering aantal treinen kan worden gebruikt in verband met de beperkte hoeveelheid geheugenruimte van de te gebruiken rekenmachine en niet in het minst de ongeveer exponentiëel toenemende benodigde rekentijd, is naast het branch-and-bound algorithme een zogenaamd koppelmechanisme ontwikkeld dat treinen, die bijna of geheel op elkaar aansluiten tot één trein reduceert. Dit stelt ons in staat grote aantallen treinen terug te brengen tot een aantal dat we met ons branch-and-bound algorithme aankunnen. Ook van dit koppel-

mechanisme is een ALGOL 60 tekst opgenomen. Overbodig zij nog op te merken dat bij gebruik van het koppelmechanisme het resultaat nog slechts suboptimaal is.

Verband met andere problemen.

De bepaling van het minimum aantal benodigde lokomotieven is een probleem dat mathematisch equivalent is met het transportprobleem, mits de onderhoudsbeurten buiten beschouwing gelaten worden. Wanneer de onderhoudsbeurten erbij betrokken worden, kan het voornoemde probleem gezien worden als een generalisatie van het handelsreizigersprobleem, door te veronderstellen dat niet één maar meerdere handelsreizigers een bepaalde hoeveelheid steden moeten bezoeken en tijdens hun tocht een aantal malen moeten terugkeren naar hun oorspronkelijk uitgangspunt. Voor het geval dat het om één handelsreiziger gaat geven C.E. Miller etc. [1] een integer programming formulering. Pogingen tot generaliseren van deze formulering leverden geen resultaat op.

## 2. Probleemstelling

We gaan uit van een zeker aantal gegeven ritten (treinen) n, die volledig bepaald worden door

a. het tijdstip van vertrek
b. het tijdstip van aankomst
c. de plaats van vertrek
d. de plaats van aankomst

Deze n ritten moeten elke dag een keer gereden worden. (We houden geen rekening met complicaties als weekenddiensten etc.)

De 'tijdsafstand' tussen twee ritten is de tijdsduur die verloopt tussen de aankomst van de eerste en het vertrek van de tweede rit. Deze tijdsafstand wordt uitgedrukt in eenheden van tien minuten. Bij het bepalen van de tijdsafstand dient rekening gehouden te worden met het feit dat iedere lokomotief tenminste 40 minuten voor het vertrek op het station

van vertrek aanwezig dient te zijn. Dit betekent dat deze tijdsafstand groter of gelijk moet zijn aan (de tijd om naar het station van vertrek te komen) + 4. Is dit niet het geval dan kan de rit pas de volgende dag plaatsvinden dus moet de tijdsafstand met 24 uur = 144 eenheden vermeerderd worden.

We vormen twee matrices $(A_{ij})^n$ en $(B_{ij})^n$ waarbij de tijdsafstand tussen de ritten in A is uitgedrukt en het aantal leegloopkm dat gereden moet worden in B, en wel op dezelfde manier als in een gewone afstandstabel. Dus:

$A_{ij}$ is de tijdsafstand van rit i naar rit j

$B_{ij}$ is het aantal leegloopkm dat gereden moet worden om van aankomstplaats rit i naar vertrekplaats rit j te komen.

N.B. A noch B is symmetrisch.

We tellen nu bij elk element van A de tijdsduur van de rit op die met de rij-index van dat element correspondeert. We bereiken hiermee dat, wanneer we een reeks ritten $r_1,r_2,\ldots,r_p$ achtereenvolgens laten verlopen en daarna terugkeren naar de vertrekplaats van rit $r_1$ (om b.v. dezelfde serie nog eens te rijden) we de tijd die daarvoor nodig is, kunnen bepalen door de elementen $A_{r_1r_2},A_{r_2r_3},A_{r_3r_4},\ldots,A_{r_pr_1}$ op te tellen.

N.B. Dit aantal is voor elke cyclus $r_1,r_2,\ldots,r_p,r_1$ een veelvoud van 144.

Met de matrix A bedoelen we verder deze nieuwe matrix.

Wanneer we nu even geen rekening houden met de onderhoudsbeurten dan betekent dat, dat het bepalen van het minimum aantal lokomotieven gebeuren kan door voor A het toewijzingsprobleem op te lossen. D.w.z.

$$\text{minimaliseer } z = \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} A_{ij}$$

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

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

$$x_{ij} = 0 \quad \text{of} \quad x_{ij} = 1\ ;$$

Het bepalen van het minimum aantal leegloopkm gebeurt door op te lossen:

$$\text{minimaliseer } y = \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} B_{ij}$$

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

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

$$x_{ij} = 0 \quad \text{of} \quad x_{ij} = 1\ ;$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} A_{ij} = z\ ;$$

We vinden als oplossing een matrix X die beschouwd kan worden als een permutatie $(i_1,i_2,\ldots,i_n)$ van de getallen $1,2,\ldots,n$. Deze permutatie is het product van een aantal cyclische permutaties:

$$(i_1,\ldots,i_n) = (i_{p_0},\ldots,i_{q_0})(i_{p_1},\ldots,i_{q_1})\ldots(i_{p_t},\ldots,i_{q_t}).$$

Zo'n cyclische permutatie $(i_{p_j},\ldots,i_{q_j})$ vertegenwoordigt een deelroute (of serie):

$$i_{p_j} \rightarrow \ldots \rightarrow i_{q_j} \rightarrow i_{p_j}$$

Nu betrekken we de revisiebeurten in ons probleem.

Om een lokomotief een onderhoudsbeurt te laten ondergaan, moet hij naar de revisiewerkplaats gereden worden. De hierbij afgelegde km zijn leegloopkm. Evenzo de km die hij moet afleggen om na de revisie weer een nieuwe rit te beginnen. Deze revisiebeurten kunnen als imaginaire

treinen opgevat worden. De tijdstippen van vertrek en aankomst vallen dan samen evenals de plaatsen van vertrek- en aankomst. De laatsten zijn dan de plaats waar de werkplaats is gevestigd. Economischer is het om de treinen die in de buurt van de werkplaats komen als 'revisierit' aan te merken, d.w.z. de lokomotief die zulk een rit aflegt wordt vóór, tijdens of ná die rit afgelost door een lokomotief die zojuist gereviseerd is. Dit houdt in dat er geen leegloopkm naar en van de werkplaats worden gemaakt.

Wanneer we nu eisen dat eens in de m dagen een lokomotief gereviseerd moet worden, dan betekent dat, dat er voor N lokomotieven $\frac{N}{m} = k$ revisieritten nodig zijn.
We willen dus k (ongeveer) even lange series vormen, met in elke serie een revisierit. Als k niet geheel is, zullen we k naar boven af moeten ronden. (Dit houdt in dat sommige series 144 eenheden = één lokomotief korter of langer zijn dan andere. Dit is geen wezenlijke complicatie voor ons probleem).

De mathematische formulering van het probleem luidt nu:

Gegeven twee matrices $(A_{ij})^n$ en $(B_{ij})^n$, een getal k en een getal N.

Gevraagd: een permutatie van de getallen 1,2,...,n die een vermenigvuldiging is van k cyclische permutaties met in elk van hen één der getallen 1,2,...,k zodanig dat voor iedere cyclus geldt:

$$A_{i_1 i_2} + A_{i_2 i_3} + \ldots + A_{i_p i_1} \leq 144 \times N/k \quad \text{en waarvoor}$$

de som van $B_{i_1 i_2} + B_{i_2 i_3} + \ldots + B_{i_p i_1}$ van de k cycli minimaal is.

## 3. Oplossingsmethode aan de hand van een voorbeeld

Door Little etc. [2] is in 1963 een branch-and-bound algorithme ontwikkeld voor de oplossing van het handelsreizigersprobleem. Prof. Kruseman Aretz [3] heeft in een voordracht dit algorithme geprogrammeerd in ALGOL 60.

Veel technieken die hierbij werden gebruikt zullen ook in dit rapport ter sprake komen.

De oplossingsmethode zal gedemonstreerd worden aan de hand van een 5-ritten voorbeeld. Het gekozen voorbeeld is niet zozeer geschikt om de snelheid van de methode aan te geven dan wel de werking van het algorithme.
De bedoelde 5 ritten zijn de volgende: (achtereenvolgens rangnummer, ritnummer, vertrekplaats, vertrektijd, aankomsttijd, aankomstplaats)

| | | | | | |
|---|---|---|---|---|---|
| 1. | 729 | Maastricht | 13.50 - | 14.20 | Heerlen |
| 2. | 317 | Elst | 9.20 - | 10.30 | Venlo |
| 3. | 100 | Amersfoort | 0.00 - | 0.30 | Utrecht |
| 4. | 116 | Deventer | 4.00 - | 4.30 | Zutphen |
| 5. | 241 | Hoek van Holland | 9.00 - | 9.40 | Zwijndrecht |

De ritten 1 en 2 zijn revisieritten. We kiezen N = 2 (het minimum) en m = 1, k is dan 2. De matrices A en B komen er nu als volgt uit te zien: (de elementen van B zijn uitgedrukt in eenheden van tien kilometers)

$$A = \begin{bmatrix} 144 & 1117 & 61 & 85 & 115 \\ 1027 & 144 & 88 & 112 & 142 \\ 83 & 56 & 144 & 24 & 54 \\ 59 & 32 & 120 & 144 & 30 \\ 29 & 146 & 90 & 114 & 144 \end{bmatrix} \qquad B = \begin{bmatrix} 2 & 114 & 20 & 20 & 25 \\ 107 & 7 & 13 & 13 & 20 \\ 18 & 7 & 2 & 8 & 8 \\ 18 & 4 & 6 & 2 & 16 \\ 21 & 12 & 8 & 14 & 4 \end{bmatrix} \quad (1)$$

sA = 0 bound = 288 sB = 0

De routes 1 → 2 en 2 → 1 mogen niet gekozen worden, daar twee revisieritten niet in dezelfde serie mogen zitten. Dus $A_{12}$ en $A_{21}$, $B_{12}$ en $B_{21}$ werden zodanig opgehoogd dat ze niet meer voor keuze in aanmerking komen. (A-elementen met 1000, B-elementen met 100). We zullen nu uit elke rij en uit elke kolom van A één element (en het daarmee corresponderende van B) moeten kiezen. Dit betekent dat als we een rij of een kolom met een bepaald bedrag wijzigen de totaalsom met datzelfde bedrag gewijzigd wordt. De optimale route blijft dus optimaal. Een en ander geldt uiteraard ook voor B. Twee problemen waarvan de afstandsschema's uit elkaar

verkregen worden door (herhaalde) kolom- of rij-veranderingen zullen we equivalent noemen.

We zullen nu (1) herleiden tot een equivalent probleem, door eerst de rijen en daarna de kolommen van A te reduceren, d.w.z. eerst in elke rij alle elementen te verminderen met het kleinste element uit die rij en vervolgens in elke kolom alle elementen te verminderen met het kleinste element uit die kolom:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 83 | 1054 | 0 | 24 | 54 | 24 | |
| | 939 | 54 | 0 | 24 | 54 | 24 | |
| A = | 59 | 30 | 120 | 0 | 30 | 30 | sA = 234 (2) |
| | 29 | 0 | 90 | 114 | 0 | 0 | bound = 54 |
| | [0] | 115 | 61 | 85 | 115 | 61 | |
| | 29 | 30 | 0 | 24 | 30 | | |

De waarde van sA geeft aan hoeveel in totaal alle routes van het nieuwe equivalente probleem korter zijn dan die van het oorspronkelijke probleem. De waarde van bound is het bedrag dat alle (nog) te kiezen elementen uit A samen mogen vormen om het maximum van twee lokomotieven = 288 eenheden niet te overschrijden. Dit bedrag bound gebruiken we om B zoveel mogelijk te reduceren. Dit doen we op dezelfde manier als bij A doch laten alle elementen die corresponderen met elementen uit A die groter zijn dan bound, buiten beschouwing.
We krijgen zo de volgende reductie:

| | | | | | | |
|---|---|---|---|---|---|---|
| | -18 | 94 | 0 | 0 | 4 | |
| | 100 | 0 | 6 | 6 | 12 | |
| B = | 11 | 0 | -5 | 1 | 0 | sB = 60 (3) |
| | 14 | 0 | 2 | -2 | 11 | |
| | [0] | -9 | -13 | -7 | -18 | |

sB heeft voor B dezelfde betekenis als sA voor A.
We hebben dus al een ondergrens voor het aantal te rijden leegloopkm.

We zullen nu een traject i → j gaan uitkiezen waarvan het voordelig lijkt het in een route op te nemen om de lengte klein te houden. Daarom laten we de keuze vallen op een traject met lengte 0 in (2), zodanig dat de som van het op één na kleinste element in kolom j en het op één na kleinste element in rij i zo groot mogelijk is. Zijn er meerdere elementen die hetzelfde presteren, dan kiezen we dat element dat de kleinste bijdrage uit B levert. Als we dan n.l. achteraf onderzoeken wat de implicaties zijn van het niet opnemen van traject i → j, dan zal de boete die we voor het weren van het traject in kwestie moeten betalen zo groot mogelijk zijn. Het is daartoe handig, achter elke rij en onder elke kolom van (2) dit op één na kleinste element te noteren. Als we de matrix A rijgewijs afzoeken valt de keuze op het omblokte element.
We krijgen nu een vertakking; we moeten van twee groepen routes de optimale gaan bepalen en deze onderling vergelijken.

De groepen zijn:

1) alle routes die niet traject 5 → 1 bevatten; minimale lengte A
   234 + 90 = 324
2) alle routes die wel traject 5 → 1 bevatten; minimale lengte A
   234

We gaan eerst groep 2) onderzoeken.
$B_{51}$ = 0 zodat de keuze van 5 → 1 geen konsekwenties heeft voor sB. Aangezien we de elementen van rij 5 en kolom 1 van beide matrices niet meer nodig hebben verwisselen we de $1^e$ en $5^e$ kolom. Dit leidt tot de volgende matrices:

| | 5 | 2 | 3 | 4 | 1 |
|---|---|---|---|---|---|
| 1 | 54 | 1054 | 0 | 24 | 83 |
| 2 | 54 | 54 | 0 | 24 | 939 |
| 3 | 30 | 30 | 120 | 0 | 59 |
| 4 | 0 | 0 | 90 | 114 | 29 |
| 5 | 115 | 115 | 61 | 85 | 0 |

sA = 234

| | 5 | 2 | 3 | 4 | 1 |
|---|---|---|---|---|---|
| 1 | 4 | 94 | 0 | 0 | -18 |
| 2 | 12 | 0 | 6 | 6 | 100 |
| 3 | 0 | 0 | -5 | 1 | 11 |
| 4 | 11 | 0 | 2 | -2 | 14 |
| 5 | -18 | -9 | -13 | 7 | 0 |

(4)

sB = 60

Daar de ritten 1 en 2 niet bij elkaar in één serie mogen zitten en we het traject 5 → 1 hebben uitgekozen, moeten de trajecten 2 → 5 en 1 → 2 onmogelijk gemaakt worden. (Voor de laatste was dit al gebeurd, doch dat is geen probleem). We hogen dus $A_{25}$ en $A_{12}$, $B_{25}$ en $B_{12}$ op. Om te weten waar deze elementen staan, hebben we naast elke rij en boven elke kolom het juiste plaatsnummer geschreven. We hoeven nu nog slechts 4 trajecten te kiezen uit het linkerbovendeel van de matrix. Noch in A, noch in B zijn in dit gedeelte nieuwe rij- of kolomreducties mogelijk. We krijgen dus:

| | 5 | 2 | 3 | 4 | 1 | |
|---|---|---|---|---|---|---|
| 1 | 54 | 2054 | 0 | 24 | 83 | 24 |
| 2 | 1054 | 54 | 0 | 24 | 939 | 24 |
| 3 | 30 | 30 | 120 | [0] | 59 | 30 |
| 4 | 0 | 0 | 90 | 114 | 20 | 0 |
| 5 | 115 | 115 | 61 | 85 | 0 | |
| | 30 | 30 | 0 | 24 | | |

| | 5 | 2 | 3 | 4 | 1 | |
|---|---|---|---|---|---|---|
| 1 | 4 | 194 | 0 | 0 | -18 | |
| 2 | 112 | 0 | 6 | 6 | 100 | |
| 3 | 0 | 0 | -5 | [1] | 11 | (5) |
| 4 | 11 | 0 | 2 | -2 | 14 | |
| 5 | -18 | -9 | -13 | -7 | 0 | |

sA = 234 bound = 54 sB = 60

We gaan nu uit de deelmatrix een traject kiezen op dezelfde manier als zo straks. De keuze valt op 3 → 4 en we krijgen een nieuwe vertakking. Te onderzoeken zijn de groepen routes:

1) alle routes die niet 5 → 1 bevatten; lengte A minstens 324
21) alle routes die wel 5 → 1 bevatten maar niet 3 → 4; minimum lengte A 288
22) alle routes die zowel 5 → 1 als 3 → 4 bevatten; minimum lengte A 234

We gaan verder met het onderzoek van groep 22.
De bijdrage van B van 3 → 4 is 1. Dus minimum lengte B wordt 61. Om $B_{3 \to 4}$ nul te maken verminderen we rij 3 met 1, dus sB wordt 61. De elementen van de rijen 3 en de kolommen 4 kunnen nu verder buiten beschouwing gelaten worden. We verwisselen daarom de rijen 3 en 4. We

krijgen:

| | 5 | 2 | 3 | 4 | 1 |
|---|---|---|---|---|---|
| 1 | 54 | 2054 | 0 | 24 | 83 |
| 2 | 1054 | 54 | 0 | 24 | 939 |
| 3 | 0 | 0 | 90 | 114 | 29 |
| 4 | 30 | 30 | 120 | 0 | 59 |
| 5 | 115 | 115 | 61 | 85 | 0 |

| | 5 | 2 | 3 | 4 | 1 |
|---|---|---|---|---|---|
| 1 | 4 | 194 | 0 | 0 | -18 |
| 2 | 112 | 0 | 6 | 6 | 100 |
| 3 | 11 | 0 | 2 | -2 | 14 |
| 4 | -1 | -1 | -6 | 0 | 10 |
| 5 | -18 | -9 | -13 | -7 | 0 |

(6)

sA = 234 sB = 61

Intermezzo

De 'classe' van een traject is het rangnummer van de rit met het kleinste rangnummer dat in een serie zit waarin ook dit traject opgenomen is. B.V. de classe van 7 → 9 in 7 → 9 → 2 → 8 is 2. Als de classe groter is dan k, dan betekent dat, dat er nog geen revisieritten in de desbetreffende serie zijn opgenomen. Dan hoeven er ook geen elementen opgehoogd te worden.

Einde intermezzo

De classe van 3 → 4 is 3, dus er hoeft niets opgehoogd te worden.

We gaan nu op dezelfde wijze als voorheen rij- en kolomreducties uitvoeren maar beschouwen daartoe alleen de 3x3 links-boven-deelmatrix. Aan matrix A blijkt niets te reduceren; kolom 1 van matrix B kan echter met 4 verminderd worden. We trekken dus 4 af van kolom 1, maar doen dat in de héle kolom teneinde een met het oorspronkelijk probleem equivalente matrix over te houden. (op de opgehoogde elementen na).
Wat we krijgen is het volgende:

| | 5 | 2 | 3 | 4 | 1 | |
|---|---|---|---|---|---|---|
| 1 | 54 | 2054 | [0] | 24 | 83 | 54 |
| 2 | 1054 | 54 | 0 | 24 | 939 | 54 |
| 4 | 0 | 0 | 90 | 114 | 29 | 0 |
| 3 | 30 | 30 | 120 | 0 | 59 | |
| 5 | 115 | 115 | 61 | 85 | 0 | |
| | 54 | 54 | 0 | | | |

| | 5 | 2 | 3 | 4 | 1 |
|---|---|---|---|---|---|
| 1 | 0 | 194 | [0] | 0 | -18 |
| 2 | 108 | 0 | 6 | 6 | 100 |
| 4 | 7 | 0 | 2 | -2 | 14 |
| 3 | -5 | -1 | -6 | 0 | 10 |
| 5 | -22 | -9 | -13 | -7 | 0 |

(7)

sA = 234 bound = 54 sB = 65

We mogen nog slechts trajecten kiezen uit het 3x3 deel. Het blijkt dat alle nul-elementen uit dit deel van A dezelfde voorkeur hebben. We kijken daarom welk element de kleinste B-bijdrage geeft. De keuze valt op 1 → 3 want deze geeft een bijdrage 0. We hadden ook 4 → 2 kunnen kiezen, daar de bijdrage hiervan ook 0 is.
We moeten nu onderzoeken de groepen routes:

1) niet 5 → 1; lengte A minstens 324
21) wel 5 → 1, niet 3 → 4; lengte A minstens 288
221) wel 5 → 1, wel 3 → 4, niet 1 → 3; lengte A minstens 288
222) wel 5 → 1 → 3, wel 3 → 4; lengte A minstens 234

We gaan verder met groep 222. We verwisselen rij 1 en rij 3. De bijdrage van B van 1 → 3 was 0, dus we houden over:

| | 5 | 2 | 3 | 4 | 1 |
|---|---|---|---|---|---|
| 4 | 0 | 0 | 90 | 114 | 29 |
| 2 | 1054 | 54 | 0 | 24 | 939 |
| 1 | 54 | 2054 | 0 | 24 | 83 |
| 3 | 30 | 30 | 120 | 0 | 59 |
| 5 | 115 | 115 | 61 | 85 | 0 |

sA = 234

| | 5 | 2 | 3 | 4 | 1 |
|---|---|---|---|---|---|
| 4 | 7 | 0 | 2 | -2 | 14 |
| 2 | 108 | 0 | 6 | 6 | 100 |
| 1 | 0 | 194 | 0 | 0 | -18 |
| 3 | -5 | -1 | -6 | 0 | 10 |
| 5 | -22 | -9 | -13 | -7 | 0 |

sB = 65 (8)

De classe van 1 → 3 is 1. We hebben al gevonden een serie 5 → 1 → 3 → 4. We moeten nu ophogen de trajecten 4 → 2 en 2 → 5. Vervolgens worden A en B weer gereduceerd (waarbij alleen de linker-boven-2x2-deelmatrix in beschouwing wordt genomen) en:

| | 5 | 2 | 3 | 4 | 1 | |
|---|---|---|---|---|---|---|
| 4 | [0] | 1000 | 90 | 114 | 29 | 1000 |
| 2 | 2000 | [0] | -54 | -30 | 885 | 2000 |
| 1 | 54 | 2054 | 0 | 24 | 83 | |
| 3 | 30 | 30 | 120 | 0 | 59 | |
| 5 | 115 | 115 | 61 | 85 | 0 | |
| | 2000 | 1000 | | | | |

sA = 288 bound = 0

| | 5 | 2 | 3 | 4 | 1 |
|---|---|---|---|---|---|
| 4 | [0] | 93 | -5 | -9 | 7 |
| 2 | 208 | [0] | 6 | 6 | 100 |
| 1 | 0 | 194 | 0 | 0 | -18 |
| 3 | -5 | -1 | -6 | 0 | 10 |
| 5 | -22 | -9 | -13 | -7 | 0 |

sB = 72 (9)

We hebben nu geen andere keus meer dan 4 → 5 en 2 → 2. We krijgen dus een route 1 → 3 → 4 → 5 → 1 en 2 → 2 met lengte A = 288 en lengte B = 72. Doordat we nu een een - naar we hopen - goede route hebben gekozen, mogen we verwachten dat de nog te onderzoeken groepen betrekkelijk weinig werk zullen leveren, daar we de ondergrenzen van A en B zo groot mogelijk hadden gekozen.
Om programmeertechnische redenen verwisselen we nog de rijen 4 en 2 en de kolommen 5 en 2. We onderzoeken verder groep 221.

In groep 221 is het traject 1 → 3 uitgesloten. We moeten dit traject door ophoging onmogelijk maken, terwijl de ophogingen van 4 → 2 en 2 → 5 ongedaan gemaakt moeten worden.
We gaan nu A en B weer vereenvoudigen, waarbij we dus weer zorgen dat in de 3x3 deelmatrices in iedere rij en in iedere kolom weer een nul staat en in de deelmatrices geen negatieve elementen voorkomen, met uitzondering van die elementen van B waarvoor het corresponderende element van A groter dan bound is.
Het resultaat:

| | 2 | 5 | 3 | 4 | 1 | |
|---|---|---|---|---|---|---|
| 2 | 54 | 1054 | 0 | 24 | 939 | 54 |
| 4 | 0 | 0 | 90 | 114 | 29 | 0 |
| 1 | 2000 | [0] | 946 | -30 | 29 | 946 |
| 3 | 30 | 30 | 120 | 0 | 59 | |
| 5 | 115 | 115 | 61 | 85 | 0 | |
| | 54 | 0 | 90 | | | |

sA = 288 bound = 0

| | 2 | 5 | 3 | 4 | 1 | |
|---|---|---|---|---|---|---|
| 2 | -6 | 102 | 0 | 0 | 94 | |
| 4 | 0 | 7 | 2 | -2 | 14 | |
| 1 | 194 | [0] | 100 | 0 | -18 | (10) |
| 3 | -1 | -5 | -6 | 0 | 10 | |
| 5 | -9 | -22 | -13 | -7 | 0 | |

sB = 71

Het meest in aanmerking komende element is nu 1 → 5. We splitsen groep 221 in twee groepen, te weten:

2211) wel 5 → 1, wel 3 → 4, niet 1 → 3, niet 1 → 5; lengte A minstens 288 + 946 = 1234

2212) wel 5 → 1 → 5, wel 3 → 4; lengte A minstens 288

We gaan verder met groep 2212.

N.B. Met de keuze van 1 → 5 is een cyclus gevormd (n.l. 5 → 1 → 5). In het algemeen moet nu gecontroleerd worden of in deze cyclus wel een revisierit is opgenomen en tevens moet nagegaan worden of de cyclus niet te lang of te kort is qua lokomotievenaantal.

I.v.m. de eenvoud van ons probleem levert bovenstaande controle geen verbod op van de gekozen deelroute.
Hoewel de classe van 1 → 5 gelijk is aan 1 hoeven we niets op te hogen daar alle kolommen en rijen waar een rangnummer 5 of 1 naast of boven staat buiten de deelmatrix vallen. (Na de verwisseling die we nu uitvoeren!)

| | 2 | 3 | 5 | 4 | 1 |
|---|---|---|---|---|---|
| 2 | 54 | [0] | 1054 | 24 | 939 |
| 4 | [0] | 90 | 0 | 114 | 29 |
| 1 | 2000 | 946 | 0 | -30 | 29 |
| 3 | 30 | 120 | 30 | 0 | 59 |
| 5 | 115 | 61 | 115 | 85 | 0 |

sA = 288

| | 2 | 3 | 5 | 4 | 1 |
|---|---|---|---|---|---|
| 2 | -6 | [0] | 102 | 0 | 94 |
| 4 | [0] | 2 | 7 | -2 | 14 |
| 1 | 194 | 100 | 0 | 0 | -18 |
| 3 | -1 | -6 | -5 | 0 | 10 |
| 5 | -9 | -13 | -22 | -7 | 0 |

sB = 71 (11)

In de 2x2 deelmatrices valt er niets te reduceren en we houden als enig mogelijke trajecten over 2 → 3 en 4 → 2. Daar lengte B = 71 hebben we een nieuwe, kortere route gevonden n.l.
1 → 5 → 1 en 2 → 3 → 4 → 2. Lengte A = 288, lengte B = 71.

Om de gekozen trajecten op de hoofddiagonaal te krijgen verwisselen we de rijen met rangnummer 2 en 4 en dat geeft het volgende beeld:

| | 2 | 3 | 5 | 4 | 1 |
|---|---|---|---|---|---|
| 4 | 0 | 90 | 0 | 114 | 29 |
| 2 | 54 | 0 | 1054 | 24 | 939 |
| 1 | 2000 | 946 | 0 | -30 | 29 |
| 3 | 30 | 120 | 30 | 0 | 59 |
| 5 | 115 | 61 | 115 | 85 | 0 |

sA = 288

| | 2 | 3 | 5 | 4 | 1 |
|---|---|---|---|---|---|
| 4 | 0 | 2 | 7 | -2 | 14 |
| 2 | -6 | 0 | 102 | 0 | 94 |
| 1 | 194 | 100 | 0 | 0 | -18 |
| 3 | -1 | -6 | -5 | 0 | 10 |
| 5 | -9 | -13 | -22 | 7 | 0 |

sB = 71 (12)

Daar alle routes uit groep 2211 een lengte A hebben van minstens 1234 kunnen we deze groep verder buiten beschouwing laten.

We gaan verder met groep 21. Dat betekent dat we de ophoging van 1 → 3 ongedaan moeten maken en het traject 3 → 4 onmogelijk moeten maken. Nu weer reduceren: (In de 4x4 deelmatrix)

| | 2 | 3 | 5 | 4 | 1 | |
|---|---|---|---|---|---|---|
| 4 | [0] | 90 | 0 | 90 | 29 | 0 |
| 2 | 54 | 0 | 1054 | 0 | 939 | 0 |
| 1 | 2054 | 0 | 54 | 0 | 83 | 0 |
| 3 | 0 | 90 | 0 | 970 | 29 | 0 |
| 5 | 115 | 61 | 155 | 85 | 0 | |
| | 0 | 0 | 0 | 0 | | |

sA = 288 bound = 0

| | 2 | 3 | 5 | 4 | 1 |
|---|---|---|---|---|---|
| 4 | [0] | 2 | 7 | -2 | 14 |
| 2 | -6 | 0 | 102 | 0 | 94 |
| 1 | 194 | 0 | 0 | 0 | -18 |
| 3 | 4 | -1 | 0 | 105 | 15 |
| 5 | -9 | -13 | -22 | -7 | 0 |

sB = 66 (13)

De keuze valt op traject 4 → 2. Te onderzoeken vallen nu nog de groepen:

1) niet 5 → 1; lengte A minstens 324
211) wel 5 → 1, niet 3 → 4, niet 4 → 2; lengte A minstens 288
212) wel 5 → 1, niet 3 → 4, wel 4 → 2; lengte A minstens 288

De classe van 4 → 2 is 2. Dus er moet opgehoogd worden en wel de trajecten 1 → 4 en 2 → 5. We zorgen dat het laatst gekozen traject op de plaats van de rechterbenedenhoek van de 4x4 deelmatrix terechtkomt

| | 4 | 3 | 5 | 2 | 1 |
|---|---|---|---|---|---|
| 3 | 970 | 90 | 0 | 0 | 29 |
| 2 | [0] | 0 | 2054 | 54 | 939 |
| 1 | 1000 | 0 | 54 | 2054 | 83 |
| 4 | 90 | 90 | 0 | 0 | 29 |
| 5 | 85 | 61 | 115 | 115 | 0 |

sA = 288

| | 4 | 3 | 5 | 2 | 1 |
|---|---|---|---|---|---|
| 3 | 105 | -1 | 0 | 4 | 15 |
| 2 | [0] | 0 | 202 | -6 | 94 |
| 1 | 100 | 0 | 0 | 194 | -18 |
| 4 | -2 | 2 | 7 | 0 | 14 |
| 5 | -7 | -13 | -22 | -9 | 0 |

sB = 66 (14)

Reduceren haalt nu niets uit dus we kiezen direct een nieuw traject en wel 2 → 4. Daar we weer een cyclus 2 → 4 → 2 gevonden hebben moeten we

eerst weer controleren of deze cyclus aan alle eisen voldoet. Dit blijkt het geval te zijn, zodat we groep 212 splitsen:

2121) wel 5 → 1, niet 3 → 4, niet 2 → 4, wel 4 → 2; lengte A minstens 288 + 970 = 1255

2122) wel 5 → 1, niet 3 → 4, wel 4 → 2, wel 2 → 4; lengte A minstens 288

Er hoeft niets opgehoogd te worden en we zorgen dat het traject 2 → 4 in de rechter-benedenhoek van de 3x3 deelmatrix terechtkomt:

| | 5 | 3 | 4 | 2 | 1 |
|---|---|---|---|---|---|
| 3 | [0] | 90 | 970 | 0 | 29 |
| 1 | 1054 | [0] | 1000 | 2054 | 83 |
| 2 | 2054 | 0 | 0 | 54 | 939 |
| 4 | 0 | 90 | 90 | 0 | 29 |
| 5 | 115 | 61 | 85 | 115 | 0 |

sA = 288

| | 5 | 3 | 4 | 2 | 1 |
|---|---|---|---|---|---|
| 3 | [0] | -1 | 105 | 4 | 15 |
| 1 | 0 | [0] | 100 | 194 | -18 |
| 2 | 202 | 0 | 0 | -6 | 94 |
| 4 | 7 | 2 | -2 | 0 | 14 |
| 5 | -22 | -13 | -7 | -9 | 0 |

sB = 66 (15)

De enig overblijvende keuze is die van de trajecten 3 → 5 en 1 → 3. Daarmee hebben we een kortere route gevonden dan we al hadden n.l. 1 → 3 → 5 → 1 en 2 → 4 → 2. Lengte A = 288; lengte B = 66.

Een nader onderzoek van groep 2121 is niet nodig want de minimum lengte A is 1255. We gaan verder met groep 211.
Daartoe moeten we de verhogingen van 1 → 4 en 2 → 5 ongedaan maken en het traject 4 → 2 ophogen. Reductie van de beide matrices geeft:

| | 5 | 3 | 4 | 2 | 1 |
|---|---|---|---|---|---|
| 3 | 0 | 90 | 970 | 0 | 29 |
| 1 | 1054 | 0 | 0 | 2054 | 83 |
| 2 | 1054 | 0 | 0 | 54 | 939 |
| 4 | 0 | 90 | 90 | 1000 | 29 |
| 5 | 115 | 61 | 85 | 115 | 0 |

sA = 288 bound = 0

| | 5 | 3 | 4 | 2 | 1 |
|---|---|---|---|---|---|
| 3 | 0 | -1 | 105 | 0 | 15 |
| 1 | 0 | 0 | 0 | 190 | -18 |
| 2 | 102 | 0 | 0 | -10 | 94 |
| 4 | 0 | -5 | -9 | 87 | 7 |
| 5 | -22 | -13 | -7 | -9 | 0 |

sB = 77 (16)

Nu ontvangen we de beloning voor het nijver uitrekenen van sB. De grens van 66 is overschreden dus groep 211 komt niet voor verdere analyse in aanmerking.

N.B. In de praktijk zal blijken dat juist deze sB voor grotere problemen als het mes optreedt waarmee vele groepen routes afgesneden worden! Dit in tegenstelling tot wat in dit voorbeeld vertoond werd waar n.l. de grens sA het leeuwendeel voor zijn rekening nam.

Blijft nog over groep 1. Ook hier is de maximum lengte A al overschreden, dus we hebben de optimale oplossing gevonden:

Twee lokomotieven rijden samen 66 leegloopkm per dag door te kiezen de route

$1 \rightarrow 3 \rightarrow 5 \rightarrow 1$ en

$2 \rightarrow 4 \rightarrow 2$

4. ALGOL 60 programma voor de methode uit 3 met toelichting.

De kern van het programma wordt gevormd door de procedure 'optimize'. Het hierdoor beschreven proces heeft de volgende externe specificaties: 'optimize' kent en werkt met een aantal grootheden, die buiten 'optimize' gedefinieerd worden. Hiertoe behoren m, het aantal ritten, en de mxm-matrices A, B en C. Aan het begin van een uitvoering van het proces bevatten A en B de tijdsafstand- en de leegloopkm-tabellen van een met het oorspronkelijk probleem equivalent probleem, behoudens dat:

a) rijen en kolommen gepermuteerd mogen zijn,
b) sommige elementen opgehoogd kunnen zijn om de keuze van de corresponderende trajecten te voorkomen.

Bij het einde van de uitvoering van elk proces voldoen A en B aan dezelfde voorwaarden. In het bijzonder zal voor elk traject, waarvan het element tijdens het proces opgehoogd is, deze ophoging weer ongedaan gemaakt worden.

```
 0 'NEDO NSloc'
 1 begin comment locomotiefdienstregelingprobleem NS ;
 2   integer n;
 3   n:= READ;
 4   begin integer i, j, k;
 5     real t;
 6     integer array route[1:n];
 7     integer array A, B, C[1:n,1:n];
 8
 9     integer procedure locomotiefdienst(A, B, C, m, route); value m;
10     integer m; integer array A, B, C, route;
11     begin integer i, j, bound, rowmini, sA, sB, r, t, lengthA,
12       lengthB, giant;
13       integer array row, col, invrow, invcol[1:m];
14       integer array rowmin, colmin[1:m];
15
16       procedure optimize(n); value n; integer n;
17       begin integer lA, ii, jj, u, v, sgn, classe;
18         NLCR; SPACE((m - n) × 5); ABSFIXT(2, 0, n);
19         for i:= 1 step 1 until n do
20         begin u:= A[i,1];
21           for j:= 2 step 1 until n do
22           begin t:= A[i,j]; if t < u then u:= t end;
23           if u ≠ 0 then
24           begin sA:= sA + u;
25             for j:= 1 step 1 until m do A[i,j]:= A[i,j] - u
26           end
27         end;
28         for j:= 1 step 1 until n do
29         begin u:= A[1,j]; v:= A[2,j]; if u > v then
30           begin t:= u; u:= v; v:= t end;
31           for i:= 3 step 1 until n do
32           begin t:= A[i,j]; if t < u then
33             begin v:= u; u:= t end
34             else if t < v then v:= t
35           end;
36           if sA > lengthA then goto end; if u ≠ 0 then
37           begin sA:= sA + u;
38             for i:= 1 step 1 until m do A[i,j]:= A[i,j] - u
39           end;
40           colmin[j]:= v - u
41         end;
42         bound:= lengthA - sA;
43         for i:= 1 step 1 until n do
44         for j:= 1 step 1 until n do if A[i,j] > bound then
45         B[i,j]:= B[i,j] + giant;
```

Welke permutaties op rijen en kolommen zijn toegepast sinds het begin van het gehele programma, is te vinden in de m vectoren 'row', 'col', 'invrow' en 'invcol'. Hierbij geeft b.v. 'row[i]' aan, welke rit bij de i-de rij van A en B hoort, terwijl 'invrow[i]' de rij van A en B aanwijst die met de i-de rit correspondeert; 'row' en 'invrow' zijn dus inverse permutaties.

Aan het begin van de uitvoering van 'optimize' zijn reeds (m-n) trajecten uitgekozen. Welke dit zijn is af te lezen uit de laatste (m-n) elementen van de vectoren 'row' en 'col'. Zo is de recentste keuze p → q geweest met p = row[n+1], q = col[n+1]; als gevolg van die keuze is juist een verwisseling uitgevoerd om de elementen corresponderende met trajecten p → x naar de n+1-ste rij, en die corresponderende met trajecten y → q naar de n+1-ste kolom te verhuizen.

Na die (m-n) reeds gedane keuzes mag 'optimize' nog n trajecten kiezen. De elementen, die hiermee corresponderen, bevinden zich juist in het n x n linker-boven gedeelte van A en B. Het is de taak van 'optimize' om

a) na te gaan of er, gegeven de (m-n) reeds gedane keuzes, en eventueel gegeven een aantal trajecten die niet gekozen mogen worden, een route te vinden is, waarvoor de lengte van A, uitgedrukt in de grootheid 'lengthA' een gegeven waarde niet overschrijdt en waarvoor de lengte van B, uitgedrukt in de grootheid 'lengthB', kleiner is dan lengthB,

b) in dat geval een bij de gegeven restricties optimale route vast te leggen in de vector 'route', en de lengte van B van die route vast te leggen in de grootheid 'lengthB'.

In de loop van de uitvoering van het programma zal 'lengthB' een monotoon dalende grootheid zijn in die zin, dat iedere wijziging een verlaging betekent. De initiële waarde van 'lengthB' zal groter dan elke route moeten zijn.

Het gehele proces wordt op gang gebracht door de activering van 'optimize' die te vinden is op regel 189 in de declaratie van het proces 'locomotiefdienst'. De procedure 'locomotiefdienst' heeft o.a. als taak de groot-

```
46          for i:= 1 step 1 until n do
47          begin u:= B[i,1];
48            for j:= 2 step 1 until n do
49            begin t:= B[i,j]; if t < u then u:= t end;
50            if u ≠ 0 then
51            begin sB:= sB + u;
52              for j:= 1 step 1 until m do B[i,j]:= B[i,j] - u
53            end
54          end;
55          for j:= 1 step 1 until n do
56          begin u:= B[1,j];
57            for i:= 2 step 1 until n do
58            begin t:= B[i,j]; if t < u then u:= t end;
59            if u ≠ 0 then
60            begin sB:= sB + u;
61              for i:= 1 step 1 until m do B[i,j]:= B[i,j] - u
62            end
63          end;
64          for i:= 1 step 1 until n do
65          for j:= 1 step 1 until n do if A[i,j] > bound then
66          B[i,j]:= B[i,j] - giant; if sB ≥ lengthB then goto end;
67          for i:= 1 step 1 until n do
68          begin u:= A[i,1]; v:= A[i,2]; if u > v then
69            begin t:= u; u:= v; v:= t end;
70            for j:= 3 step 1 until n do
71            begin t:= A[i,j]; if t < u then
72              begin v:= u; u:= t end
73              else if t < v then v:= t
74            end;
75            rowmin[i]:= v
76          end;
77          lA:= - 1;
78          for i:= 1 step 1 until n do
79          begin rowmini:= rowmin[i];
80            for j:= 1 step 1 until n do if A[i,j] = 0 then
81            begin t:= rowmini + colmin[j]; u:= lA; if t > lA then
82              begin lA:= t; ii:= i; jj:= j end;
83              if t = u then
84              begin if B[i,j] < B[ii,jj] then
85                begin ii:= i; jj:= j end
86              end
87            end
88          end;
89          t:= B[ii,jj]; if sB + t ≥ lengthB then goto end;
90          if t ≠ 0 then
91          begin sB:= sB + t;
92            for i:= 1 step 1 until m do B[ii,i]:= B[ii,i] - t
93          end;
94          u:= row[ii]; v:= col[jj]; if ii ≠ n then
```

heden 'sA', 'sB', 'lengthA', 'lengthB', 'bound', 'row', 'col', 'invrow', 'invcol' alsmede enkele andere hulpgrootheden te introduceren. Verder worden de grootheden 'row', 'col', 'invrow' en 'invcol' passend geïnitialiseerd met de identieke permutatie, en de trajecten die een verbinding leggen tussen revisieritten onderling (dit zijn de eerste k ritten) verboden door bij de corresponderende elementen van A en B een groot getal genaamd 'giant' op te tellen. Tevens wordt aan 'lengthB' datzelfde grote getal toegekend, en 'lengthA' krijgt de waarde van het product van 144 en het aantal lokomotieven dat we in de strijd wensen te werpen (183-188).*) Vervolgens wordt 'optimize' geactiveerd met als parameter m; er zijn nog geen trajecten gekozen en er zijn alleen de trajecten verboden, die revisieritten onderling verbinden. Aan alle beginvoorwaarden voor een aanroep van 'optimize' is voldaan; volgens bovenstaande specificaties zal na voltooiïng van deze aanroep de vector 'route' een optimale route aangeven, met lengte van B 'lengthB'. Daarmee is de gewenste route gevonden.

We moeten nog controleren dat de procedure 'optimize' inderdaad aan de specificaties voldoet. We gaan na wat er achtereenvolgens gebeurt. (de procedure 'optimize' bezet regel (16-181) van het programma)

Op de regels (19-27), (28-41) en (67-76) wordt de matrix A gereduceerd, zodanig dat in het n x n linkerboven-gedeelte slechts elementen groter of gelijk 0 voorkomen, terwijl hierin in iedere (deel-)rij of (deel-)kolom tenminste één 0 voorkomt. Tevens wordt de waarde van het één-na-kleinste element van iedere (deel-)rij of (deel-)kolom in de vectoren 'rowmin' en 'colmin' genoteerd. Als tijdens de kolomreductie blijkt, dat de waarde van 'sA' na de reductie groter dan 'lengthA' zal zijn, wordt het proces verlaten en is 'optimize' klaar.

In de regels (43-45) worden de elementen van B die corresponderen met dié elementen van A die groter zijn dan bound, tijdelijk opgehoogd, zodat ze bij de komende reductie van B geen rol zullen spelen.

---

*) Twee getallen met een verbindingsstreep en tussen haakjes geplaatst geven verder de regels aan in het programma waar de bijbehorende opmerkingen op doelen.

```
95          begin t:= row[n]; row[ii]:= t; row[n]:= u; invrow[u]:= n;
96            invrow[t]:= ii;
97            for j:= 1 step 1 until m do
98            begin t:= A[ii,j]; A[ii,j]:= A[n,j]; A[n,j]:= t;
99              t:= B[ii,j]; B[ii,j]:= B[n,j]; B[n,j]:= t;
100           end
101         end;
102         if jj ≠ n then
103         begin t:= col[n]; col[jj]:= t; col[n]:= v; invcol[v]:= n;
104           invcol[t]:= jj;
105           for i:= 1 step 1 until m do
106           begin t:= A[i,jj]; A[i,jj]:= A[i,n]; A[i,n]:= t;
107             t:= B[i,jj]; B[i,jj]:= B[i,n]; B[i,n]:= t;
108           end
109         end;
110         if n = 2 then
111         begin if (A[2,1] = 0 ∧ A[1,2] = 0) ∧ (B[2,1] + B[1,2] <
112           B[1,1]) ∧ (sB + B[2,1] + B[1,2] < lengthB) then
113           begin route[row[1]]:= col[2]; route[row[2]]:= col[1];
114             for t:= 3 step 1 until m do route[row[t]]:= col[t];
115             lengthB:= sB + B[1,2] + B[2,1]; lengthA:= sA;
116             ABSFIXT(3, 0, lengthA); ABSFIXT(3, 0, lengthB);
117             PRINTTEXT(⟨via:⟩); ABSFIXT(2, 0, row[2]);
118             ABSFIXT(2, 0, col[1]); PRINTTEXT(⟨and⟩);
119             ABSFIXT(2, 0, row[1]); ABSFIXT(2, 0, col[2])
120           end
121           else if sB + B[1,1] < lengthB then
122           begin lengthA:= sA; lengthB:= sB + B[1,1];
123             for t:= 1 step 1 until m do route[row[t]]:= col[t];
124             ABSFIXT(3, 0, lengthA); ABSFIXT(3, 0, lengthB);
125             PRINTTEXT(⟨via:⟩); ABSFIXT(2, 0, row[2]);
126             ABSFIXT(2, 0, col[2]); PRINTTEXT(⟨and⟩);
127             ABSFIXT(2, 0, row[1]); ABSFIXT(2, 0, col[1])
128           end
129         end n = 2
130         else
131         begin ABSFIXT(3, 0, sA); ABSFIXT(3, 0, sB); PRINTTEXT(
132           ⟨via:⟩); ABSFIXT(2, 0, u); ABSFIXT(2, 0, v); ii:= u;
133           jj:= v; lA:= lA + sA; classe:= ii;
134           for t:= invcol[ii] while t > n do
135           begin if row[t] < classe then classe:= row[t];
136             ii:= row[t]
137           end;
138           if ii ≠ jj then
139           begin if jj < classe then classe:= jj; sgn:= 1;
140             for t:= invrow[jj] while t > n do
141             begin if col[t] < classe then classe:= col[t];
142               jj:= col[t]
143             end
144           end
145           else if classe ≤ k then goto cycle else goto end;
```

De reductie van B heeft plaats op de regels (46-54) en (55-63). Bij deze reductie doen we hetzelfde als bij A alleen zijn we nu niet geïnteresseerd in de op één-na-kleinste elementen.

Ook als sB groter of gelijk dreigt te worden aan 'lengthB' tijdens de reductie, wordt het proces verlaten (regel 66), echter niet voordat de elementen van B, die in de bound-alinea opgehoogd zijn, weer hun oude waarde teruggekregen hebben (64-66).

In (77-88) wordt het meest gunstige traject bepaald. D.w.z. we zoeken een 'ii' en een 'jj' met 'ii' en 'jj' kleiner of gelijk aan n, zodanig dat A[ii,jj] = 0 of lA = rowmin[ii] + colmin[jj] is maximaal. Zijn er meerdere dan beslist de waarde van het corresponderende element van B. De boete die gelegd wordt op het niet kiezen van dit traject wordt juist vastgelegd in de locale grootheid lA.

In (89-93) wordt nagegaan of de keuze van het traject nog een bijdrage van B levert; zoja, dan wordt eerst nagegaan of de grens van 'lengthB' niet wordt overschreden (want dan kunnen we het proces beëindigen); is dit laatste niet het geval, dan trekken we van rij ii het bedrag van B ii, jj af en verhogen sB met hetzelfde bedrag.

Het gekozen traject wordt nu u → v genoemd (regel 94); door rijen en/of kolommen te verwisselen wordt bereikt dat de n-de rij met 'u' en de n-de kolom met 'v' overeenstemt (94-109).

Nu gaan we kijken of de waarde van n soms 2 is. Zoja, dan worden de regels (110-129) in behandeling genomen; zonee, dan kijken we naar (131-179). Als n=2 dan kijken we of de twee elementen die niet op de diagonaal staan misschien betere resultaten geven dan de keuze van de twee diagonaalelementen (overigens een zeldzaam verschijnsel); en, indien dat toch het geval is, of deze resultaten beter zijn dan we al met een andere route hebben gevonden. Wordt op al deze vragen positief geantwoord dan hebben we een nieuwe, voorlopig beste, route gevonden (111-120).

```
146         if classe < k then
147         for t:= 1 step 1 until k do if t ≠ classe then
148         begin i:= t;
149           for r:= invrow[i] while r > n ∧ col[r] ≠ t do i:= col[r];
150           A[invrow[i],invcol[ii]]:= A[invrow[i],invcol[ii]] + giant;
151           B[invrow[i],invcol[ii]]:= B[invrow[i],invcol[ii]] + giant;
152           j:= t;
153           for r:= invcol[j] while r > n ∧ row[r] ≠ t do j:= row[r];
154           A[invrow[jj],invcol[jj]]:= A[invrow[jj],invcol[j]] + giant;
155           B[invrow[jj],invcol[j]]:= B[invrow[jj],invcol[j]] + giant
156         end;
157         goto repeat;
158       cycle: sgn:= 0; t:= col[invrow[ii]]; r:= C[ii,t]; ii:= t;
159         for i:= invrow[ii] while col[i] ≠ t do
160         begin r:= r + C[ii,col[i]]; ii:= col[i] end;
161         if r ≠ lengthA / k then goto end;
162       repeat: optimize(n - 1); if classe < k ∧ sgn = 1 then
163         for t:= 1 step 1 until k do if t ≠ classe then
164         begin i:= t;
165           for r:= invrow[i] while r > n ∧ col[r] ≠ t do i:= col[r];
166           A[invrow[i],invcol[ii]]:= A[invrow[i],invcol[ii]] - giant;
167           B[invrow[i],invcol[ii]]:= B[invrow[i],invcol[ii]] - giant;
168           j:= t;
169           for r:= invcol[j] while r > n ∧ row[r] ≠ t do j:= row[r];
170           A[invrow[jj],invcol[jj]]:= A[invrow[jj],invcol[j]] - giant;
171           B[invrow[jj],invcol[j]]:= B[invrow[jj],invcol[j]] - giant
172         end;
173         if lA < lengthA then
174         begin A[n,n]:= A[n,n] + giant; B[n,n]:= B[n,n] + giant;
175           optimize(n);
176           A[invrow[u],invcol[v]]:= A[invrow[u],invcol[v]] - giant;
177           B[invrow[u],invcol[v]]:= B[invrow[u],invcol[v]] - giant
178         end
179       end n > 2 ;
180     end: NLCR; SPACE((m - n) × 5); PRINTTEXT(‹exit›)
181     end optimize ;
182
183     giant:= lengthB:= READ; sA:= sB:= 0; lengthA:= READ;
184     for i:= 1 step 1 until m do
185     begin row[i]:= col[i]:= invrow[i]:= invcol[i]:= i end;
186     for i:= 1 step 1 until k do
187     for j:= 1 step 1 until k do if i ≠ j then
188     begin A[i,j]:= A[i,j] + giant; B[i,j]:= B[i,j] + giant end;
189     optimize(m); NLCR; NLCR; NLCR; locomotiefdienst:= lenghtB
190   end locomotiefdienst ;
191
192   for i:= 1 step 1 until n do
193   for j:= 1 step 1 until n do A[i,j]:= C[i,j]:= READ;
194   for i:= 1 step 1 until n do
195   for j:= 1 step 1 until n do B[i,j]:= READ; k:= READ;
```

Is het antwoord op één van deze vragen echter negatief, dan kijken we of de route, vertegenwoordigd door de diagonaalelementen, gunstiger is dan de tot dusver best gevondene. Is dit het geval, dan hebben we een nieuwe route gevonden en dat wordt vastgelegd in de vector 'route'. In ieder geval is voor n=2 het proces beëindigd (121-128).

Als n groter is dan 2 dan gaan we verder op regel 131. Eerst leggen we de ondergrens vast van de groep routes waarin het traject u → v verboden is. Dit doen we met lA. Vervolgens wordt de classe van dit traject bepaald (132-144). Tevens werd nagegaan of door de keuze van het traject u → v een cyclus werd gevormd. Is dit niet het geval, dan wordt aan de locale variabele 'sgn' de waarde 1 toegekend. Werd er wel een cyclus gevormd, dan wordt eerst gekeken of de cyclus wel een revisierit bevat ('classe' moet dan kleiner zijn dan k); zonee, dan wordt het proces beeindigd; zoja, dan gaan we naar (158-160) om te kijken of de cyclus wel de goede lengte van A heeft. Daar dit alleen kan worden nagegaan aan de hand van het oorspronkelijke probleem gebruiken we de m x m-matrix C die we de vorm gaven van de oorspronkelijke A en waarmee we verder niets gedaan hebben. (Ook geen rij- of kolom-permutaties toegepast!) Verder krijgt 'sgn' de waarde 0.

Als er geen cyclus werd gevormd en de classe was niet groter dan k dan moet er iets opgehoogd worden. Dit gebeurt in (146-156). Hierin worden dié en alleen dié elementen (trajecten) opgezocht die een verbinding tussen twee revisieritten leggen (al of niet via andere reeds gekozen trajecten).

Hetzelfde proces dat we tot dusver gevolgd hebben voor n, gaan we nu uitvoeren voor n-1. Als er in deze tak een betere route zit, dan zal die er zeker uitrollen. Na beëindiging van het onderzoek van deze tak moeten de elementen die eventueel in (146-156) opgehoogd werden, weer verlaagd worden (162-172).

Als de ondergrens van de groep routes zonder het traject u → v groter is dan 'lengthA' dan hoeven we geen verder onderzoek meer te plegen en

```
196     ABSFIXT(6, 0, locomotiefdienst(A, B, C, n, route));
197     PRINTTEXT(‡ leegloopkm , via ‡);
198     for i:= 1 step 1 until k do
199     begin NLCR; j:= i; ABSFIXT(3, 0, i);
200       for j:= route[j] while j ≠ i do ABSFIXT(3, 0, j)
201     end
202   end
203 end
204
```

is 'optimize(n)' voltooid. Deze ondergrens was vastgelegd in '1A'. Was '1A' niet groter dan 'lengthA' dan moeten we eerst de tak onderzoeken waarin het traject u → v verboden is (174-178). Daartoe worden eerst de elementen van A en B die corresponderen met u → v opgehoogd en onderzoeken we d.m.v. 'optimize(n)' of er in de overgebleven n x n-matrix misschien betere routes te vinden zijn. Daarna worden de ophogingen weer ongedaan gemaakt en is het proces beëindigd!

In de rest van het programma worden de matrices A, B en C alsmede enige andere probleemafhankelijke grootheden ingelezen en een aantal output-opdrachten verstrekt, teneinde de optimale route en de weg naar deze route te laten printen.

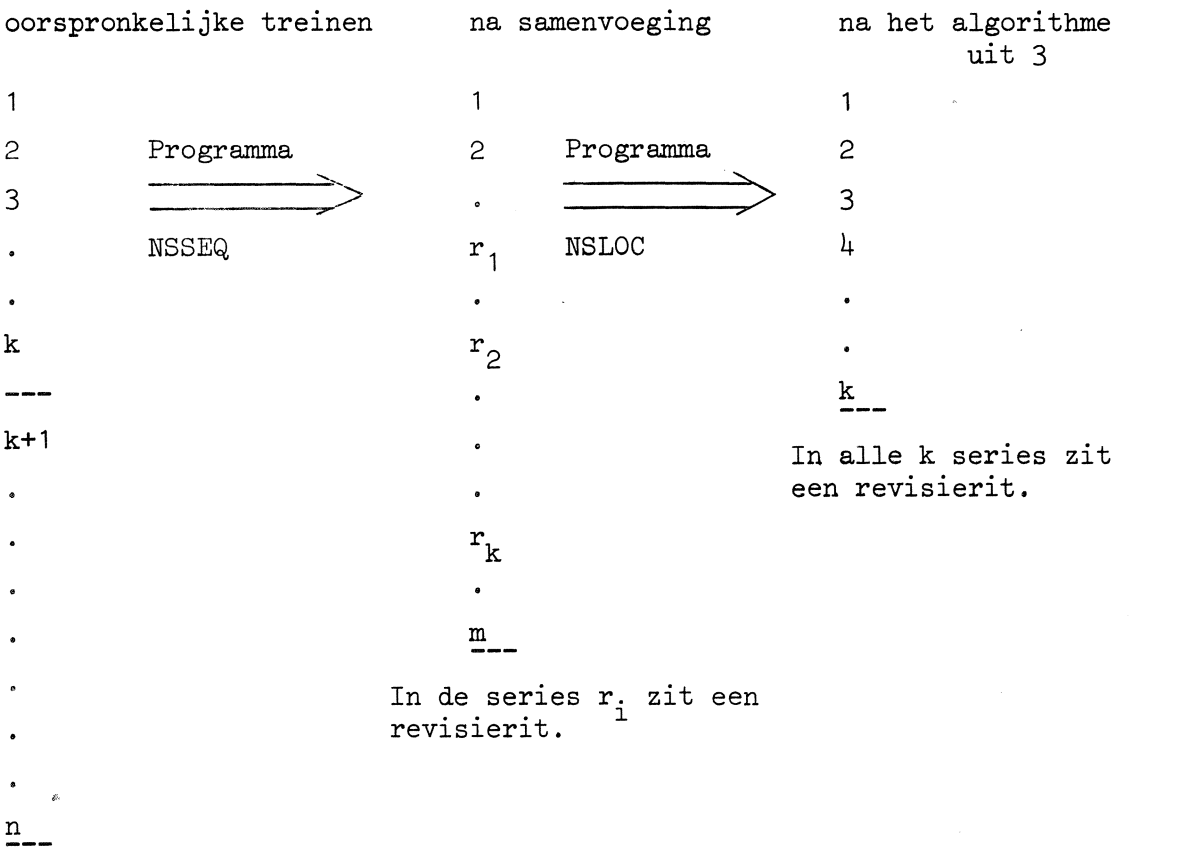
Het in 3 besproken voorbeeld levert de volgende output:

```
  5  234   60 VIA:  5   1
       4  234   61 VIA:  3   4
            3  234   65 VIA:  1   3
                 2  288   72 VIA:  4   5 AND  2   2
               EXIT
            3  288   71 VIA:  1   5
                 2  288   71 VIA:  2   3 AND  4   2
               EXIT
          EXIT
          EXIT
       4  288   66 VIA:  4   2
            3  288   66 VIA:  2   4
                 2  288   66 VIA:  3   5 AND  1   3
               EXIT
          EXIT
       4
     EXIT
     EXIT
     EXIT
EXIT

     66  LEEGLOOPKM , VIA
   1    3    5
   2    4
```

5. Methode om grote problemen om te zetten in kleine.

De hieronder beschreven methode is bedoeld als ad-hoc methode die het probleem dat schrijver dezes heeft willen oplossen rijp maakt voor het branch-and-bound algorithme uit hoofdstuk drie. Zij is zodanig provisorisch dat er nog geen enkele pretentie omtrent optimaliteit aan ten grondslag ligt. Wel zal het mogelijkerwijs het raam aangeven, waarbinnen een algemeen bruikbare methode ontwikkeld kan worden. In het kort kan zij als volgt worden omschreven:
Zij gegeven n ritten, waarvan de eerst k revisieritten zijn. Voeg nu die treinen, die bijna of geheel op elkaar aansluiten samen, echter zodanig dat twee revisieritten niet in dezelfde serie terecht kunnen komen. Dit samenvoegen kan gebeuren door steeds zwakkere criteria te stellen tot we een aantal treinen m overhouden. Van dit aantal noemen we die treinen waarin een revisierit is opgenomen weer revisierit; we hebben dus weer k revisieritten.
Op deze m treinen kunnen we nu de methode van hoofdstuk drie toepassen om een optimale volgorde te krijgen. In een schema kunnen we het als volgt weergeven:




Een en ander is meer gedetailleerd uitgewerkt in het programma NSSEQ in hoofdstuk 6.

## 6. ALGOL 60 programma voor de methode uit 5 met toelichting

De belangrijkste delen van het programma zijn de Boolean procedure Conditie en de cyclus die begint met LL: (resp. (9-28) en (40-61). De overige regels bevatten declaraties en inlees- en output-opdrachten.
In de vectoren 'nummer', 'vertrekpunt', 'eindpunt', 'starttijd', en 'aankomsttijd' worden de ritten vastgelegd. Deze ritten worden ingelezen op volgorde van vertrektijd. De vector 'rij' dient ervoor om de koppelingen die plaatsvinden vast te leggen. Zo zal aan het eind van de uitvoering van het programma 'rij i' aangeven welke rit is gekoppeld vóór de rit met 'nummer i'.
De Boolean vector 'keten' krijgt de waarde true als de corresponderende rit achter een andere rit wordt gekoppeld; zolang dit nog niet het geval is, houdt hij de waarde false. Een nieuwe rit, die wordt gevormd door de koppeling van twee ritten, krijgt het nummer van de eerstvertrekkende rit.
De Boolean vector 'revisierit' krijgt de waarde true als in de desbetreffende serie een revisierit is opgenomen, anders false. Aan het begin van de uitvoering van het programma krijgen dus alleen de k revisieritten een 'revisierit'-waarde true.
In de matrix 'afstand' kan de afstand tussen de verschillende stations worden uitgedrukt.
De grootheid 'L' heeft tot bedoeling een serie ritten die samen een veelvoud van 24 uur gaan overschrijden de kans te geven in de "volgende dag" nieuwe aansluitmogelijkheden te zoeken. Dit gebeurt door middel van een cyclus (regel 61).
Het proces verloopt nu als volgt:

De regels 41 en 42 worden uitgevoerd. Stel we zijn met het for-statement gevorderd tot de uitvoering voor 'j' = j en 'i' = i. 'j' is sterkte van het koppel-criterium; hoe lager de waarde van 'j' hoe sterker het criterium. 'i' geeft het rangnummer van de rit aan waarachter we een te koppelen rit zoeken. Dit doen we alleen als 'keten i' is false; was de waarde true dan werd deze rit reeds geïdentificeerd met een andere rit en zouden we door aansluiting te zoeken dubbel werk doen.

```
0     'NEDO NSseq'
1     begin integer n, a, i, j, m, k;
2       n:= READ; a:= READ;
3       begin integer l, s, t, max, f;
4         integer array nummer, vertrekpunt, eindpunt, starttijd,
5         aankomsttijd, rij[1:n], afstand[1:a,1:a];
6         Boolean array keten, revisierit[1:n];
7         Boolean b, c, d, e, g;
8
9         Boolean procedure Conditie(i, m, j, l); value i, m, j, l;
10        integer i, m, j, l;
11        begin integer t;
12          t:= starttijd[m] - aankomsttijd[i] - 4 + 1; b:= t > 0;
13          c:= t + 144 > 0; d:= t ≤ j × 2; e:= t + 144 ≤ j × 2;
14          f:= afstand[eindpunt[i],vertrekpunt[m]];
15          if ((b ∧ d) ∨ (c ∧ e)) ∧ f ≤ (j - 1) × 2 ∧ ((b ∧ f ≤ t) ∨
16          (¬b ∧ f ≤ t + 144)) then
17          Conditie:= true;
18          begin if ¬b then
19            begin if aankomsttijd[m] - starttijd[i] + 1 > 144 then
20              Conditie:= false else g:= true
21            end
22            else
23            begin if aankomsttijd[m] - starttijd[i] + 1 > 0 then
24              Conditie:= false
25            end;
26          end
27          else Conditie:= false
28        end;
29
30        l:= - 144; k:= READ; t:= 0; max:= READ;
31        for i:= 1 step 1 until n do
32        begin rij[i]:= nummer[i]:= READ; vertrekpunt[i]:= READ;
33          eindpunt[i]:= READ; starttijd[i]:= READ;
```

We gaan dus een aansluiting zoeken. Daartoe onderzoeken we de ritten die daarvoor in aanmerking komen op hun geschiktheid. Ten eerste mogen zij niet reeds in een andere serie, anders dan als eerstvertrekkende rit, zijn opgenomen. Ten tweede mogen door de keuze van de desbetreffende rit geen revisieritten worden gekoppeld (44-45). Wordt aan bovenstaande twee voorwaarden voldaan, dan wordt met behulp van de procedure Conditie nagegaan of aan alle overige aansluiteisen werd voldaan.
Krijgt de procedure Conditie de waarde false, dan wordt nagegaan of we al zover van een theoretisch mogelijke aansluittijd afzitten, dat het geen zin meer heeft om verder te zoeken. (de ritten waren immers gesorteerd op vertrektijd!). Dit gebeurt op regel 57. Zaten we nog dicht genoeg in de buurt dan proberen we de volgende rit. Waren we al ver genoeg afgedwaald, dan hoeven we voor deze waarde van 'i' niet meer te zoeken en kunnen we de volgende rit nemen om aansluitingen achter te zoeken.

Krijgt de procedure Conditie de waarde true, dan zijn er geen beletselen meer om de aansluiting een feit te doen zijn en wordt deze aansluiting uitgevoerd. Tegelijkertijd wordt geteld of met deze aansluiting het aantal ritten voldoende is gereduceerd om het branch-and-bound algorithme het verdere werk te laten doen. Dit tellen gebeurt met behulp van de grootheid 't'.

Bij het doen plaatsvinden van de aansluiting wordt met de logische variabele 'b' nagegaan of het een aansluiting van dezelfde dag betreft of dat het middernachtelijk uur werd gepasseerd. (We werken in feite modulo 144). Was het bovenstaande het geval dan moet bij de nieuwe aankomsttijd nog eens 144 bijgeteld worden.
De grootheid 'b' kreeg zijn waarde tijdens de uitvoering van de procedure 'Conditie'.

Tenslotte wordt opdracht gegeven de gevormde combinaties te printen en wordt het programma beëindigd (uiteraard alleen als 't' voldoende groot is geworden, anders gaat het proces gewoon door).

```
34          aankomsttijd[i]:= READ; keten[i]:= false;
35          revisierit[i]:= nummer[i] > 900
36        end;
37        for i:= 1 step 1 until a do
38        for j:= 1 step 1 until i do if i ≠ j then afstand[i,j]:=
39        afstand[j,i]:= READ else afstand[i,j]:= 0;
40      LL: l:= l + 144;
41        for j:= 1 step 1 until k do
42        for i:= 1 step 1 until n do if ¬keten[i] then
43        begin g:= false;
44          for m:= i step 1 until n, 1 step 1 until i do if ¬keten[m]
45          ∧ ¬(revisierit[i] ∧ revisierit[m]) then
46          begin if Conditie(i, m, j, l) then
47            begin t:= t + 1; keten[m]:= true;
48              if revisierit[m] then revisierit[i]:= true;
49              eindpunt[i]:= eindpunt[m];
50              if ¬b then aankomsttijd[i]:= aankomsttijd[m] + l + 144
51              else aankomsttijd[i]:= aankomsttijd[m] + l;
52              s:= rij[m]; rij[m]:= rij[i]; rij[i]:= s; NLCR;
53              ABSFIXT(4, 0, rij[m]); ABSFIXT(4, 0, nummer[m]);
54              if t = n - max then goto end
55            end
56            else
57            begin if (¬d ∨ (¬e ∧ g)) then goto break end;
58          end;
59        break:
60        end;
61        goto LL;
62      end: NLCR; PRINTTEXT(
63        ‡de volgende combinaties werden gevormd‡);
64        for i:= 1 step 1 until n do
65        begin NLCR; ABSFIXT(4, 0, nummer[i]); ABSFIXT(4, 0, rij[i]);
66          if ¬keten[i] then PRINTTEXT(‡ beginpunt ‡)
67        end
68      end
69    end
```

## 7. Enige resultaten.

Het programma NSLOC werd getest op het in hoofdstuk 3 behandelde probleem van 5 ritten. Eveneens werd dit programma getest op een 12-ritten probleem. De tijd die de EL X8 voor het laatste probleem nodig had was 52 sec. Hetzelfde probleem werd met de hand in 18 uur tijds opgelost. Bij dit probleem werd de oplossing gevonden, enkel door 'naar rechts te gaan', d.w.z. de eerst nagegane uiterst rechtse tak bleek achteraf de optimale te zijn geweest.

Het programma NSSEQ werd getest op een testset, verstrekt door het Mathematisch Centrum. De naam van deze testset was 38/300. Zij bestond uit 38 stations en 310 treinen. Het aantal revisieritten werd op 5 gesteld.

In het algemeen kunnen we stellen dat de tijd die nodig is om een programma van het type van NSLOC uit te voeren exponentieel toeneemt met het aantal ritten; een hypothese die voor het handelsreizigersprobleem door Little [2] wordt bevestigd. Little presteerde het om op een IBM 7090 een 40-steden probleem binnen 8 minuten op te lossen, zij het dan dat hij voor de afstanden aselect getrokken getallen koos. Het bleek dat het toevoegen van tien steden de rekentijd met een faktor tien deed vermeerderen.

Gezien dit feit, alsmede gezien de 52 sec. die de X8 nodig had om het 12-rittenprobleem op te lossen, mogen we niet verwachten dat een probleem van meer dan 20 ritten binnen redelijke tijd opgelost kan worden met het programma NSLOC alleen.

## 8. Slotopmerkingen

Tot besluit volgende hieronder nog een paar kanttekeningen:

1. Hoewel de ontwikkelde branch-and-bound methode ontstaan is met het doel grote problemen op te lossen, d.w.z. problemen waarbij een groot aantal lokomotieven en treinen een rol spelen, zal het duidelijk zijn dat juist de bij kleine problemen verkregen optimaliteit een steeds kleiner effect gaat sorteren als de problemen groter worden. Dit wordt veroorzaakt door de (nog) grove wijze van koppelen die we eerst moeten toepassen om het probleem handzaam te maken. Schrijver dezes is een ieder dankbaar, die met suggesties komt om tot een verantwoorde schaalverkleining te komen. Een schaalverkleining dus, waarbij het nuttig effect van een toepassing van het branch-and-bound algorithme ook voor grotere problemen nog bewaard blijft.

2. Om tot een schatting te komen hoeveel tijd de oplossing van een probleem vergt, kunnen we mogelijkerwijs het programma uit [3] gebruiken. Dit programma dient om het handelsreizigersprobleem op te lossen. Daartoe moeten we twee zaken vergelijken n.l. de structuren van beide programma's en het theoretisch aantal mogelijkheden waaruit een keuze zal moeten worden gedaan.
   De structuren van beide programma's zijn ongeveer gelijk, zij het dat ons programma met twee matrices werkt en het programma van Kruseman Aretz met één. We kunnen grof stellen dat dit een faktor twee in rekentijd uitmaakt.
   Het theoretisch aantal mogelijkheden van het handelsreizigersprobleem is $(n-1)!$ vooropgesteld dat we met een niet-symmetrisch geval van doen hebben.
   Het bepalen van het theoretisch aantal mogelijkheden van ons probleem is een minder voor de hand liggende zaak. De vraag is: Op hoeveel manieren kan men n-k ritten over k plaatsen verdelen, waarbij dan nog de volgorde op elk van die k plaatsen een rol speelt.

Om dit probleem op te lossen, kiezen we k getallen $i_1,i_2,\ldots,i_k$ met $i_1 + i_2 + \ldots + i_k = n - k$.
Hebben we deze getallen $i_h$ vast gekozen, dan vragen we ons af op hoeveel manieren we n-k ritten van de tekens 1,2,...,k kunnen voorzien en waarbij we telkens $i_h$ tekens h gebruiken.
Dit is juist $\frac{(n-k)!}{i_1!i_2!\ldots i_k!}$ manieren.
Daar elke $i_h$ ritten die het teken h krijgen op $i_h!$ verschillende volgordes geplaatst kunnen worden moeten we bovenstaande uitdrukking met $i_1!i_2!\ldots i_k!$ vermenigvuldigen, zodat we vinden (n-k)! verschillende manieren.
Nu hadden we één manier bekeken waarbij de getallen $i_1,i_2,\ldots$ en $i_k$ gefixeerd waren. We moeten dus tellen op hoeveel manieren (verschillend) deze getallen $i_1,i_2,\ldots,i_k$ kunnen worden gekozen. Voor k = 1 is dat maar op één manier: alle n-k elementen moeten naar vakje 1. Voor k = 2 zijn er n-k+1 manieren n.l.

| vakje 1 | vakje 2 |
|---|---|
| 0 | n-k |
| 1 | n-k+1 |
| . | . |
| . | . |
| . | . |
| . | . |
| n-k | 0 |

Als we algemeen $f(k,n-k)$ het aantal manieren noemen waarop de k getallen $i_1,i_2,\ldots,i_k$ kunnen worden gekozen, dan luidt de oplossing van het probleem $(n-k)!f(k,n-k)$.
Van de functie $f$ weten we het volgende:

$f(1,n-k) = 1$ voor alle n-k

$f(k,n-k) = f(k-1,n-k)+f(k-1,n-k-1)+f(k-1,n-k-2)+\ldots+f(k-1,0)$

Na enig nadenken en de driehoek van Pascal nog eens goed bekeken te hebben komen we tot de slotsom dat:

$f(k,n-k) = \binom{n-1}{n-k}$ , zodat het antwoord op het probleem wordt:

$$(n-k)!f(k,n-k) = (n-k)!\binom{n-1}{n-k} = \frac{(n-1)!}{(k-1)!} .$$

Terugkerend tot de vergelijking met het handelsreizigersprobleem zien we dat dat probleem precies (k-1)! maal zoveel theoretische mogelijkheden heeft. Daar ruw gezegd, de hoeveelheid rekenwerk per mogelijkheid half zo groot was, verwachten we dat de oplossing van het handelsreizigersprobleem ongeveer $\frac{1}{2}(k-1)!$ zoveel werk (rekentijd) vraagt als de oplossing van ons probleem van dezelfde grootte.

## 9. Literatuurverwijzingen

[1] 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

[2] 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

[3] Kruseman Aretz, F.E.J., "Waarom langer reizen dan nodig is?", voordracht in de serie "Elementaire onderwerpen vanuit hoger standpunt belicht". Uitgave van de Stichting M.C.