NETWERKPLANNINGTECHNIEKEN II

Tijdbepaling en netwerkplanning

door Drs. A. Bosman en Drs. K. Wezeman

4 Inleiding

Nadat in de eerste aflevering van deze serie artikelen over het onderwerp netwerkplanningtechnieken de algemene grondslagen aan de orde zijn gesteld, zal in dit tweede artikel worden ingegaan op één van de varianten van deze technieken, nl. de critical path scheduling (CPS)[1]. Wij zullen in de volgende paragraaf in het bijzonder aan de tijdbepaling en de daaruit resulterende instrumenten aandacht besteden. Vervolgens zullen een tweetal methodes worden besproken die kunnen worden gebruikt bij de tijdbepaling, nl. een methode voor het „met de hand" doorrekenen van een netwerk en een methode die kan worden gebruikt indien men met behulp van een elektronische rekenmachine de noodzakelijke berekeningen wil uitvoeren.

5 De tijdbepaling bij „Critical Path Scheduling"

Met CPS wordt die variant van de netwerkplanningtechnieken bedoeld, waarbij men uitgaat van gedetermineerde tijdsduren der verschillende in het netwerk voorkomende activiteiten. Met behulp van deze tijdsduren en de opeenvolgingsrelatie der activiteiten bepaalt men het kritieke pad en derhalve ook de tijd - t_n^l - noodzakelijk voor de uitvoering van het plan. Het resultaat van deze tijdbepaling kan vervolgens worden vergeleken met een normtijd - de projectduur λ - die door de leiding kan worden vastgesteld, waarbij rekening is gehouden met de wensen van de opdrachtgever ten aanzien van het moment van oplevering. De leiding kan nu direct vaststellen of het plan aan de gestelde eisen voldoet. Dit is nl. het geval in de situatie dat $\lambda \geq t_n^l$. In het andere geval, $\lambda < t_n^l$, is het niet mogelijk het plan binnen de gestelde tijd uit te voeren, wat betekent dat er een nieuw plan, d.w.z. een nieuw netwerk, moet worden opgesteld[2]).

Welke instrumenten kunnen nu aan de netwerkplanningtechnieken voor de bedrijfsleiding worden ontleend? In de eerste plaats is dit een duidelijk criterium, nl. het vergelijken van λ met t_n^l, voor het bepalen of een plan al of niet kan worden geaccepteerd. In de tweede plaats biedt het kritieke pad de mogelijkheid om in:

a) het geval dat $\lambda < t_n^l$ direct vast te stellen van welke activiteiten de duur in eerste instantie moet worden verkort, of welke activiteiten, indien dat om welke reden dan ook niet mogelijk is, naar andere paden met een relatief grote totale speelruimte moeten worden verschoven;

---

[1]) Zie, *Maandblad voor Accountancy en Bedrijfshuishoudkunde*, jrg. 41 (1967), nr. 8.

[2]) Het is dikwijls niet mogelijk met alle factoren die op de activiteitsduur invloed kunnen uitoefenen afzonderlijk rekening te houden, alleen reeds door het simpele feit dat het verband tussen deze factoren en elke activiteitsduur niet kan worden vastgesteld. In die situatie kan het gewenst zijn t_n^l met een bepaald percentage α te verhogen en de aldus gevonden tijdsduur $t_n^l + \alpha\, t_n^l$ met λ te vergelijken.

b) het geval dat $\lambda \geqq t_n^l$ in het bijzonder aandacht te besteden aan de kritieke activiteiten - doorgaans niet veel meer dan 20% van het totaal aan activiteiten - waardoor het mogelijk wordt aan het begrip management by exception een, in planning en uitvoering, kwantificeerbare inhoud te geven.
In de derde plaats geven de diverse begrippen speelruimte de leiding de mogelijkheid in bepaalde gevallen naar een betere - soms naar een optimale - oplossing van het planningprobleem te streven. In de klassieke netwerkplanningtechnieken maakt alleen de Critical Path Method (CPM) hiervan gebruik[3]). Een voorwaarde voor het zoeken naar een dergelijke oplossing is dat naast het onder het eerste punt genoemde tijdcriterium, dat weinig selectief is, een aanvullend criterium wordt gebruikt. In het geval van CPM is dat een minimalisatie van de kosten verbonden aan de reducering van de duur van elke activiteit. Dit betekent dat dan tenminste twee activiteitsduren en de daarbij behorende kosten voor de uitvoering ervan bekend moeten zijn. Bij CPM is dit inderdaad het geval. Men onderscheidt een „normal" en een „crash time". Met behulp van lineaire programmering kan dan een optimale oplosing voor de reducering van de projectduur in een gegeven netwerk worden gevonden. Dit laatste is een groot voordeel, omdat een verkorting van de projectduur nu betekent dat geen nieuw netwerk, het tijdrovendste deel van de gehele netwerkprocedure, behoeft te worden opgesteld[4]). Langs deze weg is het mogelijk acceptabele waarden van t_n^l te vinden tegen minimale kosten voor de verandering van de activiteitsduren.

Indien men rekening gaat houden met het beslag van elk der activiteiten op produktiefactoren en de hoeveelheden van deze factoren beschikbaar in een bepaalde periode, dan ontstaat een allocatieprobleem, nl. de toerekening van de activiteiten aan deze factoren waarvoor met de beschikbare technieken geen optimale oplossing, enkele triviale gevallen buiten beschouwing gelaten, kan worden gevonden[5]). Desalniettemin zijn er programma's ontwikkeld die met behulp van heuristische principes een oplossing trachten te vinden. Een van de meest gebruikte instrumenten daarbij is het begrip speelruimte. Dit laatste ligt voor de hand, omdat de speelruimte een maatstaf geeft voor de periode waarmee een activiteit kan worden uitgesteld of verschoven zonder de lengte van de projectduur, ervan uitgaande dat $\lambda = t_n^l$, in gevaar te brengen. Van de vier, in het vorige artikel genoemde, begrippen speelruimte kunnen er twee in vele gevallen niet eenduidig per activiteit worden bepaald, nl. de totale en de afhankelijke speelruimte. Dit is een gevolg van het feit dat deze speelruimten kunnen worden toegerekend aan een aantal elkaar opvolgende activiteiten. Vindt een dergelijke toerekening plaats, geheel aan één activiteit of voor delen aan verschillende activiteiten, dan is daarmee ook de speelruimte voor de niet aan de toerekening deelnemende activiteiten verdwenen. De vrije en onafhankelijke speelruimte kunnen

---

[3]) Onder klassieke netwerkplanningtechnieken verstaan wij die technieken, als PERT, CPM, en CPS, waarbij een oplossing van het planningprobleem wordt gezocht onder de veronderstelling dat de produktiefactoren in voldoende omvang aanwezig zijn om het probleem te kunnen oplossen.

[4]) Voor een verdergaande uiteenzetting over CPM zie, A. Bosman: „Netwerkplanningtechnieken II", M.B.A., oktober/november 1965.

[5]) Aan deze problematiek zal hier verder geen aandacht worden besteed.

[6]) Voor de berekening van s^0 wordt gebruik gemaakt van de formule $s^0 = \max. (0, t_j^e - t_i^l - t_{ij})$. Dit betekent dat bij het voorkomen van een negatieve onafhankelijke speelruimte deze voor de waarde nul wordt opgenomen.

aan één activiteit worden toegekend[6]). Voor het verkrijgen van een juist inzicht in de mogelijkheden voor de toerekening van de afhankelijke en totale speelruimte aan bepaalde activiteiten is het van belang na te gaan in hoeverre deze reeds bestaan uit aan één activiteit gebonden componenten. Voor de totale speelruimte werd dit reeds aangegeven in (8) uit het eerste artikel.
Immers

$$s^t = s^v + s^a \qquad (1)$$

zodat men de s^t slechts voor de grootte van s^a kan verplaatsen. Ook tussen s^v en s^0 blijkt een bepaald verband te bestaan. Trekt men van s^v s^0 af, dan resteert

$$s_{ij}^a = t_j^l - t_i^e \qquad (2)$$

zodat geldt

$$s^v = s_{ij}^a + s^0 \qquad (3)$$

(bij het uitvoeren van deze berekening moet men de juiste waarde, dus ook de negatieve, van s^0 gebruiken).

Aangezien s^v en s^0 beide per activiteit kunnen worden vastgeseld, moet dat in dit geval ook gelden voor s_{ij}^a. Dit is in strijd met de conclusie hiervoor, nl. dat s^a niet eenduidig per activiteit zou kunnen worden bepaald. De tegenstelling schuilt in de definitie van:

$$s^v = t_j^e - t_i^e - t_{ij} \qquad (4)$$

Door hierin uit te gaan van j wordt de vrije speelruimte toegekend aan de laatste activiteit van een rij opeenvolgende activiteiten met eenzelfde s^a. Door deze afspraak over de toerekening van s^v wordt de s_{ij}^a per definitie een gefixeerde grootheid. In wezen is dus van de vier begrippen speelruimte de s^0 de enige die zonder meer aan elke activiteit kan worden toegerekend. Over het algemeen is er nog weinig aandacht besteed aan het opstellen van procedures om de totale en vrije speelruimte aan de afzonderlijke activiteiten toe te rekenen[7]). Het is echter juist daarom van belang de grootte van s^0 te kennen, omdat men hierdoor, in het geval van $s^0 > 0$, een indicatie krijgt welk deel van de totale speelruimte bij een bepaalde activiteit hoort.

Na deze algemene beschouwing over de tijdbepaling zal deze nu aan de hand van het in het voorgaande artikel beschreven getallenvoorbeeld nader worden uitgewerkt. Met behulp van de in paragraaf 3 gegeven formules kan men nu alle relevante momenten berekenen (zie figuur 1 en tabel II).

Zoals reeds werd gesteld is de projectduur - λ - gelijk aan t_n^l, in ons voorbeeld is dit 32 weken. Het kritieke pad bestaat uit de activiteiten (0,1), (1,2), (2,3), (3,4), (4,6), (6,7), (7,9), (9,11), (11,13), (13,14), (14,15), (15,18), (18,19), (19,20).

Uit de tijdbepaling der knooppunten volgt ook de omvang der verschillende speelruimten. Deze zijn weergegeven in tabel III.

---

[7]) Een uitzondering hierop is een programma samengesteld door General Electric, zie: *GE-225 Application - Critical Path Method Program*, General Electric Computer Department, Phoenix 1962. Het rapport zelf behandelt de wijze waarop deze toerekening plaats vindt niet; deze vindt men beschreven bij Karl Weber: „Planung mit CPM und PERT - Verfeinerung und Weiterentwicklung", *Industrielle Organisation*, Vol. 33 (1964), nr. 6.

Tabel II

Activiteit [8]	t_{ij}	t^e_i	t^l_i	t^e_{ij}	t^l_{ij}	Activiteit	t_{ij}	t^e_i	t^l_i	t^e_{ij}	t^l_{ij}
0,1	4	0	4	4	0	8,10	2	17	22	19	20
0,2	4	0	6	4	2	9,11	2	19	21	21	19
1,2	2	4	6	6	4	10,12	0,5	19	22,5	19,5	22
2,3	5	6	11	11	6	11,13	1	21	22	22	21
2,4	3	6	12	9	9	12,14	0	19,5	22,5	19,5	22,5
2,5	2	6	14	8	12	13,14	0,5	22	22,5	22,5	22
3,4	1	11	12	12	11	14,15	5	22,5	27,5	27,5	22,5
3,5	0	11	14	11	14	14,16	3	22,5	26,5	25,5	23,5
4,6	3	12	15	15	12	15,18	2	27,5	29,5	29,5	27,5
5,7	1	11	15	12	14	16,17	2	25,5	28,5	27,5	26,5
6,7	0	15	15	15	15	17,19	1	27,5	29,8	28,5	28,5
6,8	2	15	20	17	18	18,19	0	29,5	29,5	29,5	29,5
7,9	4	15	19	19	15	19,20	2,5	29,5	32	32	29,5

Tabel III

Activiteit	s^t	s^v	s^a	s^o	Activiteit	s^t	s^v	s^a	s^o
0,1	0	0	0	0	8,10	3	0	3	0
0,2	2	2	0	2	9,11	0	0	0	0
1,2	0	0	0	0	10,12	3	0	3	0
2,3	0	0	0	0	11,13	0	0	0	0
2,4	3	3	0	3	12,14	3	3	0	0
2,5	6	3	3	3	13,14	0	0	0	0
3,4	0	0	0	0	14,15	0	0	0	0
3,5	3	0	3	3	14,16	1	0	1	0
4,6	0	0	0	0	15,18	0	0	0	0
5,7	3	3	0	0	16,17	1	0	1	0
6,7	0	0	0	0	17,19	1	1	0	0
6,8	3	0	3	0	18,19	0	0	0	0
7,9	0	0	0	0	19,20	0	0	0	0

Uit deze tabel blijkt, dat op het kritieke pad zowel de totale als de vrije speelruimten nul zijn.

[8] Hoe de verschillende momenten worden berekend, zal in de volgende paragraaf nog nader worden uiteengezet. Men kan bij de berekening bijv. uitgaan van de activiteiten en het feit dat $t^e_i = 0$ en de respectievelijke paden in het netwerk volgen; aldus zijn de tabellen II en III samengesteld.

6 Berekeningstechnieken

De uitkomsten van de tijdbepaling zoals die in de tabellen II en III zijn opgenomen, kunnen worden berekend met behulp van het netwerk en de in paragraaf 3 gegeven formules. Voor netwerken van enige omvang is dit een zeer tijdrovende bezigheid. Het is echter mogelijk met behulp van een eenvoudig hulpmiddel het „met de hand" doorrekenen van netwerken te vereenvoudigen en te bespoedigen. Indien men de mogelijkheid heeft de berekeningen door een elektronische rekenmachine te laten uitvoeren, waarvoor deze zich zeer goed lenen, staat men voor de opgave de machine van een programma te voorzien. Bij de opstelling van een dergelijk programma is men gebonden, in negatieve zowel als in positieve zin, aan het instrument waarvan men gebruik maakt, hetgeen veelal zal betekenen dat een andere methode moet worden gevolgd dan bij een berekening „met de hand". In deze paragraaf zullen beide mogelijkheden worden besproken.

Men kan een graph op verschillende manieren in de vorm van een matrix, bestaande uit nullen en enen, weergeven[9]). Een van deze matrices zou men kunnen aanduiden met de term knooppuntmatrix. Het i,j element van deze matrix is een 1 indien er, in het geval van een gerichte graph, een pijl loopt van knooppunt i naar knooppunt j. In het andere geval heeft dit element de waarde 0. Vervangt men nu in deze matrix de enen door de activiteitsduren en zorgt men ervoor dat bij de knooppuntnummering $i < j$, dan resulteert een driehoeksmatrix, met op de hoofddiagonaal en het daaronder gelegen deel van de matrix alleen nullen. Heeft men een dergelijke matrix opgesteld, zoals in tabel IV voor ons voorbeeld, dan kunnen twee knooppuntmomenten, nl. het vroegst mogelijke en het laatst toelaatbare knooppuntmoment, worden berekend. Zoals in par. 3 reeds werd aangetoond kunnen alle andere momenten op eenvoudige wijze hieruit worden afgeleid.

[9]) Een aantal van deze matrices wordt behandeld door Robert G. Busacker en Thomas L. Saaty: *Finite Graphs and Networks*, New York 1965, zie hoofdstuk 5.

Tabel IV

t_i^e	i \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	0		4	4																		
4	1			2																		
6	2				5	3	2															
11	3					1	0															
12	4							3														
11	5								1													
15	6								0	2												
15	7										4											
17	8											2										
19	9												2									
19	10													0,5								
21	11														1							
19,5	12															0						
22	13															0,5						
22,5	14																5	3				
27,5	15																			2		
25,5	16																		2			
27,5	17																				1	
29,5	18																				0	
29,5	19																					2,5
32	20																					
t_i^l		0	4	6	11	12	14	15	15	20	19	22	21	22,5	22	22,5	27,5	26,5	28,5	29,5	29,5	32
$t_i^l - t_i^e$		0	0	0	0	0	3	0	0	3	0	3	0	3	0	0	0	1	1	0	0	0

Voor de bepaling van het vroegst mogelijke knooppuntmoment gaat men als volgt te werk. Plaats in de t_i^e kolom voor het knooppunt 0 een 0. Ga vervolgens voor het knooppuntnummer, waarvoor men t_i^e wil berekenen, uit van de rij met dit nummer. Vervolgens gaat men vanaf het snijpunt van deze rij met de kolom met hetzelfde nummer naar boven en men telt bij elke waarde die men tegenkomt de reeds bepaalde t_i^e op van de rij waarin het betreffende getal staat. Daarbij neemt men steeds als de gezochte waarde het grootste getal dat op deze wijze wordt bepaald. Dit is dan de geschatte t_i^e van het knooppunt waarvan men uitging. Zo vindt men voor knooppunt 7 een t_j^e van 15 (nl. door vergelijking van 11 + 1 en 15 + 0). De berekening van het laatst toelaatbare knooppuntmoment verloopt op omgekeerde wijze. Deze grootheid wordt immers gevonden door, uitgaande van de totale projectduur, het netwerk van achter naar voren door te rekenen. In ons voorbeeld is $\lambda = t_{20}^l = 32$ weken. Men gaat nu evenseens uit van de rij met het knooppuntnummer waarvoor de berekening moet plaats vinden. De elementen die men tegenkomt worden afgetrokken van de t_j^l in de kolom waarin dit element zich bevindt, waarbij de kleinste waarde de gezochte is. Zo vindt men de t_{14}^l door

vergelijking van 26,5 — 3 en 27,5 — 5. Nadat op de hiervoor beschreven wijze de t_i^e en de t_i^l zijn berekend, kunnen daaruit met behulp van de bekende formules de overige in tabel II opgenomen grootheden worden bepaald.

Hoewel de hier beschreven rekenmethode eenvoudig en overzichtelijk is, leent zij zich toch minder goed voor toepassingen waarbij gebruik wordt gemaakt van een rekenmachine. Hiervoor zijn twee redenen. In de eerste plaats het bezwaar dat men bij de knooppuntnummering altijd moet voldoen aan de voorwaarde $i < j$. Voor grote en complexe netwerken, waarvan de onderdelen dikwijls door verschillende instanties worden samengesteld, is dit een groot bezwaar. Om hieraan tegemoet te komen heeft men programma's samengesteld die in een netwerk de knooppunten zodanig nummeren dat aan de genoemde voorwaarde wel wordt voldaan. Bij enigszins grote netwerken kan de uitvoering van een dergelijk programma echter veel meer tijd vergen dan de gehele daarop volgende netwerkberekening. Een tweede bezwaar is, dat bij netwerken van enige omvang speciale maatregelen moeten worden getroffen om de knooppuntmatrix op te lossen. Immers in een netwerk met 100 knooppunten bevat een dergelijke matrix reeds 10.000 elementen, zodat snel de capaciteit van het interne geheugen van de rekenmachine wordt bereikt[10]).

De speciale maatregelen die men in dit geval zou kunnen nemen zijn, gebruik maken van een extern geheugen - indien althans aanwezig en voldoende snel bereikbaar - om de matrix in gedeelten, de grootte ervan afhankelijk van de capaciteit van het interne geheugen, in de machine in te lezen. Dit laatste is uiteraard alleen mogelijk indien aan de hiervoor genoemde voorwaarden - $i < j$ - wordt voldaan. Men behoeft dan per knooppunt alleen t^e en t^l vast te houden. Het merendeel van de nu bestaande programma's maakt van deze laatste mogelijkheid, in één of andere vorm, gebruik.

Men kan programma's voor de rekenmachine opstellen die beide genoemde bezwaren niet hebben. In Nederland is dat bijv. gedaan door Nederkoorn[11]). In zijn programma wordt voor de berekening van de beide momenten gebruik gemaakt van een recursieve procedure. In het algemeen kan men een dergelijke procedure omschrijven als een operatie waarin een bepaald deel van het programma door middel van een instruktie in één gang herhaalde malen wordt doorlopen en waarbij elke nieuwe fase wordt ingezet met de uitkomsten in de daaraan voorafgaande gevonden. Het attractieve van deze operatie is dat men in staat is iteratieve processen in bepaalde programmeertalen in een aantal simpele opdrachten op te schrijven. Een dergelijk iteratief proces is de berekening van de beide knooppuntmomenten, omdat de waarden ervan onderling afhankelijk zijn[12]).

De ALGOL compiler van de rekenmachine aan de Rijksuniversiteit te Groningen staat recursieve operaties niet toe. Onder gebruikmaking van het merendeel der overige bestanddelen van het programma van Nederkoorn, moesten wij dus een andere procedure opstellen voor de berekening van de beide knooppunt-

[10]) Met de momenteel ter beschikking staande machines en programma's kunnen, voorzover ons bekend, netwerken met maximaal 100.000 activiteiten worden verwerkt.

[11]) J. Nederkoorn: *A PERT program in ALGOL 60*, rapport MR 56 van de Rekenafdeling van het Mathematisch Centrum te Amsterdam, Amsterdam 1963.

[12]) Voor een uiteenzetting over de grondslagen van dit programma en de wijze waarop de recursieve operatie daar wordt toegepast, wordt men verwezen naar J. Nederkoorn: „PERT, een voorbeeld van de besliskundige aanpak", *Maandblad voor Bedrijfsadministratie en organisatie*, 69e jrg. (1965), nr. 814.

momenten. Gezien het nauwe verband dat er bestaat tussen de netwerkplanning en de graphentheorie, waarop in paragraaf 2 reeds werd gewezen, lag het voor de hand dat getracht werd gebruik te maken van de resultaten die deze speciale tak van de wiskunde reeds heeft bereikt. Wij doelen hier op de in deze tak van wiskunde bekende oplossingen van de zgn. labyrintproblemen[13]). Een formulering van het labyrinthprobleem in de taal van de graphentheorie zou als volgt kunnen luiden: in een eindige graph V worden twee knooppunten, bijv. s en z, gekozen. Gezocht wordt een pad tussen s en z. Dit pad moet een subgraph zijn van de graph V. Het is voor het vinden van een oplossing niet noodzakelijk dat men V in zijn geheel kent. Voor het oplossen van dit soort problemen is een aantal algoritmes ontwikkeld, die ondermeer worden besproken in het bekende boek van König[14]). In het programma dat in Groningen is gemaakt[15]), ter vervanging van de recursieve onderdelen in het programma van Nederkoorn, wordt gebruik gemaakt van een variant van een algoritme van Wiener[16]). Hierbij wordt uitgegaan van twee richtlijnen nl.:

a) dat, wanneer men bij de wandeling door het doolhof in een knooppunt K aankomt, voor elke naar K lopende pijl bekend is of deze reeds werd doorlopen;
b) de reeds afgelegde weg moet elk ogenblik in tegengestelde richting opnieuw kunnen worden afgelegd.

De oplossing van Wiener behelst nu het volgende. Men begint de wandeling in een willekeurig knooppunt A als beginpunt en doorloopt vervolgens de ribben AB, BC, CD . . . enz. tot men een eindpunt van de graph, of een knooppunt dat reeds werd gepasseerd, bereikt. Indien dit het geval is legt men de reeds afgelegde weg in de omgekeerde volgorde af tot een knooppunt wordt bereikt waarin een nog niet doorlopen ribbe eindigt. Deze ribbe volgend zet men de wandeling over de nog niet beschreven ribben voort tot wederom een eindpunt of een knooppunt wordt bereikt dat reeds werd beschreven; dan keert men weer om, enz. Men kan bewijzen dat men met deze methode altijd in het uitgangspunt A terugkomt, terwijl dan alle ribben van de graph zijn beschreven[17]). Dit principe is verwerkt in een procedure ter bepaling van het vroegst mogelijke startmoment der activiteiten. Deze procedure is geschreven in ALGOL en is een onderdeel van een programma waarin alle bij CPS voorkomende berekeningen t.a.v. knooppuntmomenten kunnen worden gemaakt. In de bijlage bij dit artikel is de ALGOL-tekst

---

[13]) Men kan immers de paden in een labyrint opvatten als ribben in een graph en de verbindingen tussen deze paden als knooppunten.

[14]) D. König, *Theorie der endlichen und unendlichen Graphen*, Leipzig 1936, p. 35 e.v.

[15]) De schrijvers zijn zeer veel dank verschuldigd aan Drs. H. J. Buurema van het Mathematisch Instituut der Rijksuniversiteit te Groningen, die veel heeft bijgedragen tot de totstandkoming van dit programma.

[16]) Naast de hier genoemde methode van Wiener vermeldt König nog een tweetal andere methoden voor het oplossen van labyrintproblemen, nl. die van Trémaux en die van Tarry. De methode van Trémaux is met succes toegepast bij het aftasten van de oplossingsruimte bij lineaire programmeringsproblemen, waarbij men alle sub-optimale oplossingen kan verkrijgen die een bepaald, variabel, percentage van de waardefunctie verschillen met de optimale oplossing. Zie A. Bosman en J. Mol: *Near optimality Analysis: An Application to Crop Rotation Planning*, rapport voor het ES/TIMS congres in Warschau, 1-8 september 1966.

[17]) De methode van Wiener kan in bepaalde gevallen minder efficiënt zijn dan de methoden van Trémaux en van Tarry. Bij Wiener is het nl. mogelijk dat een bepaalde ribbe meer dan tweemaal wordt doorlopen, hetgeen bij Trémaux en Tarry niet kan. De procedure toegepast in het programma-onderdeel „procedure Est ( )" is een variant van de methode van Wiener. Door gebruik te maken van het feit dat het netwerk een gerichte graph is, wordt het meer dan tweemaal doorlopen van een ribbe in het netwerk voorkomen.

opgenomen met een kleine toelichting. Uitgaande van de laatste activiteit wordt het netwerk doorlopen tot een beginpunt wordt bereikt[18]). Dit geschiedt als volgt. In het geheugen van de machine zijn opgenomen een rangnummer voor elke activiteit, de begin- en eindknooppunten ervan alsmede de tijdsduren der verschillende activiteiten, waarbij de volgorde van de invoer van de activiteiten en de nummering der knooppunten willekeurig kan zijn. Door de machine wordt nu het beginknooppunt van de laatste activiteit, aangeduid met het rangnummer ervan tussen de haken van Est ( ), opgezocht in de kolom beginknooppunten. Tevens wordt het rangnummer van deze activiteit in het geheugen, in de array Etage - zie bijlage - vastgelegd. Vervolgens wordt in de rij der eindknooppunten nagegaan of er een activiteit is die dit zelfde knooppunt als eindknooppunt heeft. Is dit het geval dan volgt daaruit dat deze activiteit aan de desbetreffende activiteit voorafgaat. In het andere geval is de conclusie dat men met een beginactiviteit te doen heeft, òf dat deze activiteit niet van belang is voor de laatste activiteit. Indien er een activiteit wordt gevonden, die aan een andere activiteit voorafgaat, dan wordt nagegaan of de vroegste start ervan reeds bekend is. Is dit niet het geval dan wordt deze activiteit achter de reeds in het geheugen geplaatste activiteiten opgenomen. Vervolgens wordt het beginknooppunt van deze activiteit opgezocht waarna wederom wordt nagegaan of er een activiteit is die dit nummer als eindknooppunt heeft. Is de vroegste start nog niet berekend dan wordt deze activiteit vervolgens in het geheugen opgenomen etc. etc. Op deze wijze ontstaat in de array Etage een reeks van activiteitnummers die de weg voorstelt waarlangs men is gekomen. Deze reeks wordt in omgekeerde volgorde teruggelopen zodra een beginpunt is bereikt. De vroegste start van de eerste activiteit wordt op 0 gesteld, van waaruit de vroegste start van activiteiten op het pad, waarlangs dit beginpunt wordt bereikt, kan worden berekend. Deze worden voor elke activiteit in een speciale kolom weggeschreven. Tijdens dit teruglopen gaat de machine opnieuw voor elke activiteit na of er nog andere activiteiten zijn die aan de desbetreffende activiteit voorafgaan. Dit vindt plaats op de reeds beschreven wijze van vergelijking van begin- en eindknooppuntnummer. Indien dit het geval is dan wordt de terugweg niet verder afgelopen, maar wordt dit zijpad ingeslagen[19]). Dit weer onder de voorwaarde dat het vroegste startmoment van deze activiteit nog niet bekend is. Ook van dit zijpad worden de rangnummers weggeschreven en wel in dezelfde rij te beginnen bij de plaats waar de machine was gebleven. Indien weer een beginpunt is bereikt, of een punt waarvan het vroegste startmoment bekend is, wordt de rij weer in omgekeerde volgorde afgebroken waarbij voor elk knooppunt wordt nagegaan of er reeds een vroegste startmoment werd bepaald. Zijn er meerdere vroegste startmomenten dan wordt in dat geval het maximum genomen. Deze gang van zaken wordt herhaald tot alle paden zijn beschreven. De omgekeerde procedure wordt gevolgd bij de berekening van het laatst toelaatbare voltooiingsmoment van elke activiteit.

---

[18]) Bij toepassing van dit programma mag een netwerk meerdere beginactiviteiten hebben, maar slechts één eindactiviteit. Hieraan zal men altijd kunnen voldoen door aan het netwerk als laatste aciviteit een schijnactiviteit te verbinden.

[19]) Gegeven het gerichte karakter van de onderhavige netwerken voert een dergelijk zijpad altijd naar een beginpunt of een punt waarvan het vroegste startmoment bekend is. Het gerichte karakter van het netwerk wordt tot uitdrukking gebracht in het onderscheid dat in het programma wordt gemaakt tussen begin- en eindknooppunt van een activiteit. Door hiervan gebruik te maken kan de methode van Wiener efficiënter worden toegepast.

Korte toelichting

D (/. .,Q/) is de aanduiding van een twee dimensionale array, die te beschouwen is als een tabel met twee ingangen, waarin de grondgegevens worden opgenomen en waarin tevens de resultaten worden weggeschreven. De array telt Q rijen voor het aantal activiteiten NJ; het aantal kolommen is een variabele grootheid, afhankelijk van de aard van het programma.

De in deze procedure voorkomende kolommen (de kolommen 1 t/m 5) herbergen de volgende gegevens:

kolom 1: de beginknooppuntnummers der activiteiten;
„ 2: de eindknooppuntnummers;
„ 3: de tijdsduren der activiteiten;
„ 4: de vroegste start van een activiteit; bij het begin van het programma wordt in deze kolom overal —1 gezet ten teken dat nog niets werd berekend;
„ 5: de vrije speelruimte der activiteiten.

Als de procedure EST (A) wordt aangeroepen is A het rangnummer van de laatste activiteit. In de eendimensionale array ETAGE wordt de afgelegde weg bijgehouden door het rangnummer van de doorlopen activiteit in deze array op te nemen, welk nummer weer uit de array wordt verwijderd als in de procedure de desbetreffende activiteit voor de tweede maal, d.w.z. in tegengestelde richting, is doorlopen.

```
 0  'BEGIN' 'INTEGER' 'PROCEDURE' EST(A)., 'VALUE' A., 'INTEGER' A.,
 1  'BEGIN' 'INTEGER' ARRAY' ETAGE(/0..NGR/).,
 2          'INTEGER' ETAGENR,Z,DIZ,R,Q,RH.,
 3          ETAGENR.=0., ETAGE(/0/).=A.,
 4  QQQ.. Z.=ETAGE(/ETAGENR/)., DIZ.=D(/1,Z/).,
 5          R.=0., 'FOR' Q.=1 'STEP' 1 'UNTIL' NJ 'DO'
 6  'BEGIN' 'IF' DIZ 'EQUAL' D(/2,Q/) 'THEN'
 7  'BEGIN' 'COMMENT' "DWZ Q GAAT AAN Z VOORAF".,
 8          'IF' D(/4,Q/) 'EQUAL' —1 'THEN'
 9  'BEGIN' 'COMMENT' "Q GAAT AAN Z VOORAF, MAAR D(/4,Q/) NOG ONBEKEND.,
10          ETAGENR.=ETAGENR+1.,
11          ETAGE(/ETAGENR/).=Q.,
12 'COMMENT' "VOOR Q WORDT EEN NIEUWE ETAGE GEBOUWD WAARIN DE BEREKENING
13          VAN D(/4,Q/) KAN PLAATS VINDEN".,
14            'GOTO' QQQ.,
15  'END' 'ELSE'
16  'BEGN' 'COMMENT' "D(/4,Q/) IS BEKEND".,
17          RH.=D(/4,Q/)+D(/3,Q/).,
18          'IF' RH 'GREATER' R 'THEN' R.=RH., D(/5,Q/).=RH
19  'END'.,
20  'END'., 'COMMENT' "ALS DIZ ONGELIJK D(/2,Q/) DAN Q NIET VAN BELANG".,
21  'END'., 'COMMENT' "VAN HET FOR-STATEMENT".,
22          D(/4,Z/).=R., 'COMMENT' "DIT TREEDT ALLEEN OP ALS ER IN HET
23          FOR-STATEMENT GEEN ETAGE VERHOGING HEEFT PLAATSGEVONDEN, D.W.Z.
24          ALS D(/4,Q/) BEKEND IS".,
25      'FOR' Q.=1 'STEP' 1 'UNTIL' NJ 'DO'
26      'IF' DIZ 'EQUAL' D(/2,Q/) 'THEN'
27      D(/5,Q/) .=R—D(/5,Q/).,
28          ETAGENR.=ETAGENR—1.,
29          'IF' ETAGENR 'NOTLESS' 0 'THEN' 'GOTO' QQQ.,
30  EST.=R.,
31  'END'., 'COMMENT' "EINDE PROCEDURE EST".,
```