stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNDE BN 20/73 AUGUSTUS

B.J. LAGEWEG

MATROIDEN EN VOLGORDEPROBLEMEN

2e boerhaavestraat 49 amsterdam

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

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), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

AMS(MOS) onderwerpen classificatie schema (1970): 90B35, 05B35

Inhoud

1. Matroïden

1.1. Inleiding 1

1.2. De greedy algorithm 4

1.3. Toepassingen 5

1.4. Definities van matroïde 9

2. Polyhedra

2.1. Polyhedra van matroïden 11

2.2. Maximale koppelingen 14

3. Een algoritme voor een optimale branching

3.1. Inleiding 18

3.2. Matroïdale formulering 18

3.3. L.p.-formulering 22

3.4. Algoritme 24

3.5. Efficiency en optimaliteit 27

Literatuur 28

Appendix: ALGOL-procedures 31

1. Matroïden.

1.1. Inleiding.

We vergelijken de volgende twee problemen:

1. het lineaire toewijzingsprobleem (LAP)

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}\, x_{ij}$$

onder:

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

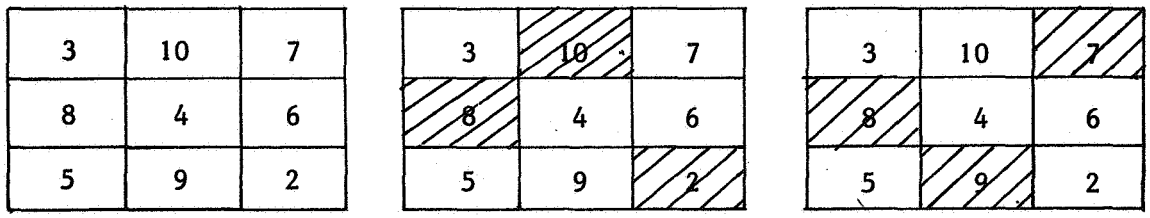
$$\sum_{i=1}^{n} x_{ij} = 1 \quad , \quad j = 1,\ldots,n \; ;$$

$$x_{ij} \in \{0,1\} \quad , \quad i = 1,\ldots,n \; , \quad j = 1,\ldots,n \; .$$

2. het minimale opspannende boomprobleem (STP):
   vind in een samenhangende niet-gerichte graaf een opspannende boom, zó dat de som van de gewichten van de kanten van de boom minimaal is.

Bij beide problemen moet uit een verzameling (vz.) elementen V, resp. van cellen (i,j) en van kanten e_j, een deelverzameling (deelvz.) gekozen worden die aan bepaalde eisen voldoet en een optimaal gewicht heeft.
We construeren als volgt een oplossing B voor probleem P.
We beginnen met B de lege vz. We kiezen een element $v \in V$ met optimaal gewicht en verwijderen v uit V.
Als $B \cup \{v\}$ een toegelaten oplossing van P is, dan wel uitgebreid kan worden tot een toegelaten oplossing van P, voegen we v aan B toe. We gaan hiermee door totdat B een oplossing van P is. Deze algoritme staat bekend als de *greedy algorithm* (GA) [11].

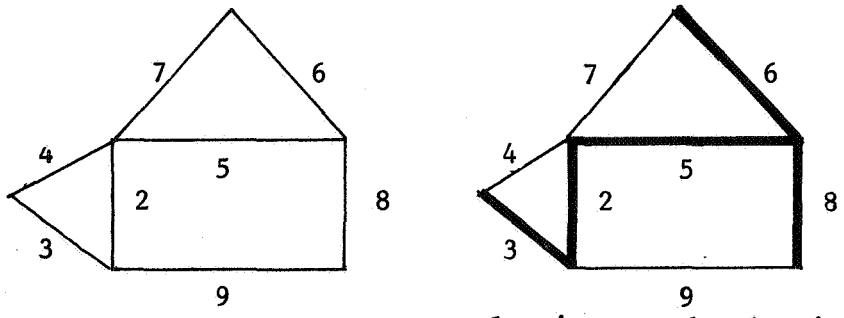
Voorbeeld 1 : LAP



oplossing greedy algorithm

optimale oplossing

Voorbeeld 2 : STP



oplossing greedy algorithm = optimale oplossing

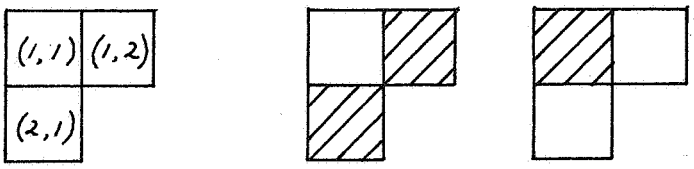
Voorbeeld 1 illustreert dat de GA niet noodzakelijk een optimale oplossing van het LAP geeft. In geval van het STP is de GA identiek aan de algoritme van KRUSKAL [22], die een optimale oplossing oplevert. Dit verschillend gedrag t.o.v. de GA kan worden verklaard d.m.v. matroïden.
Van de vele mogelijke definities van matroïde (zie 1.4) gebruiken we hier

Definitie 1. Zij E een eindige vz. elementen en F een niet-lege collectie deelvzn. van E (de zgn. onafhankelijke vzn.).
Het paar M = (E,F) is een *matroïde* indien geldt:

(1) F is een onafhankelijk stelsel vzn., dwz een deelvz. van een onafhankelijke vz. is onafhankelijk;

(2) voor elke $A \subseteq E$ bevatten alle maximaal onafhankelijke deelvzn. van A evenveel elementen.

Een maximaal onafhankelijke deelvz. van een vz. A is een onafhankelijke vz., die niet is bevat in een onafhankelijke vz. met meer elementen. Een maximaal onafhankelijke deelvz. van E heet een *basis* van de matroïde M. De *rang* r(A) van een vz. A is het aantal elementen van een maximaal onafhankelijke deelvz. A. De rang van de matroïde M is gedefinieerd als r(E), het aantal elementen van een basis van M. Een *circuit* is een minimaal onafhankelijke vz., dwz. elke eigenlijke deelvz. van een circuit is onafhankelijk.



A

twee maximaal onafhankelijke deelvz. van A

Figuur 1.

De partiële oplossingen van het LAP vormen een onafhankelijk stelsel, maar voldoen niet aan (2), zoals blijkt uit Figuur 1, maw. definiëren geen matroïde.

De collectie F van partiële oplossingen van het STP op een graaf G = (V,E), dwz. alle vzn. kanten zonder cycles (zie voor het begrip cycle § 3.1), vormen een matroïde M = (E,F), een *graafmatroïde*.

Immers, zij A een deelvz. van E, bestaande uit p disjuncte samenhangende componenten $A_1,\ldots\ldots,A_p$.

$G_i = (V_i,A_i)$ is de deelgraaf van G met als vz. punten V_i de punten van G incident met A_i. Een vz. kanten in G_i zonder cycles is maximaal d.e.s.d. als zij een opspannende boom is van G_i. Elke opspannende boom van G_i telt $|V_i|-1$ kanten. Bijgevolg heeft elke maximaal onafhankelijke deelvz. van A

$\sum_{i=1}^{p} |V_i| - p$ elementen.

In de volgende paragraaf wordt aangetoond dat de GA een optimale oplossing van een probleem P geeft als de partiële oplossingen van P de onafhankelijke vzn. van een matroïde zijn.

1.2. De greedy algorithm.

Gegeven is een matroïde $M = (E,F)$ met elementen $E = \{e_1,.....,e_n\}$. c is een reële functie op E ; we schrijven c_j voor $c(e_j)$. We veronderstellen dat de elementen van E zo genummerd zijn, dat $c_1 \geq c_2 \geq \ldots \geq c_n$.
De greedy algorithm luidt:

0. Initialiseer $A_0 = \emptyset$, $B_0 = \emptyset$ en $j = 1$.
1. $A_j = A_{j-1} \cup \{e_j\}$.

$$B_j = \begin{cases} B_{j-1} \cup \{e_j\} & , \text{ als } B_{j-1} \cup \{e_j\} \in F, \\ B_{j-1} & , \text{ anders .} \end{cases}$$

2. Als $j < n$, stel dan $j := j + 1$ en ga naar 1.

De GA construeert een vz. B_n, die een basis van M is met maximaal gewicht. Om dit te bewijzen geven we eerst twee lemma's.

Lemma 1 : Eigenschap (2) is equivalent met de zgn. aanvullingseigenschap (3):

(3) Voor elk tweetal onafhankelijke deelvzn. I_k en I_{k+1} met k, resp. k+1 elementen geldt:
er is een element $e \in I_{k+1} \setminus I_k$, zó dat $I_k \cup \{e\}$ een onafhankelijke vz. is.

Bewijs: (3) $\Rightarrow$ 2 : Als $I_k \subset A$ en $I_{k+1} \subset A$, is I_k niet maximaal.
(2) $\Rightarrow$ 3 : $r(I_{k+1} \cup I_k) \geq k+1$, omdat $I_{k+1} \subseteq I_{k+1} \cup I_k$. I_k is dus niet maximaal in $I_{k+1} \cup I_k$ en kan uitgebreid worden met een element $e \in I_{k+1} \setminus I_k$.

Lemma 2 : $r(A_j) = |B_j|$, voor $j = 0,\ldots,n$.

Bewijs : Het bewijs verloopt met volledige inductie naar j.
Voor $j = k-1$ geldt $r(A_{k-1}) = |B_{k-1}|$.
Als B een maximaal onafhankelijke deelvz. van A_k is, is $|B_{k-1}| \leq |B| \leq |B_{k-1}| + 1$, omdat B_{k-1} een onafhankelijke deelvz. van A_k is, $A_k = A_{k-1} \cup \{e_k\}$ en B_{k-1} een maximaal onafhankelijke deelvz. van A_{k-1} is.
Als $|B| = |B_{k-1}|$, is $B_{k-1} \cup \{e_k\} \notin F$ (anders $r(A_k) > |B|$), zodat de GA $B_k = B_{k-1}$ stelt en $r(A_k) = |B| = |B_{k-1}| = |B_k|$.

Als $|B| = |B_{k-1}| + 1$, bestaat o.g.v. lemma 1 een $e \in B \setminus B_{k-1}$ met $B_{k-1} \cup \{e\} \in F$. Als $e \neq e_k$ is $B_{k-1} \cup \{e\} \subseteq A_{k-1}$ en $r(A_{k-1}) \geq |B_{k-1}| + 1$.
Dus $e = e_k$ en $B_{k-1} \cup \{e_k\} \in F$, zodat
$r(A_k) = |B_{k-1}| + 1 = r(B_{k-1} \cup \{e_k\}) = r(B_k) = |B_k|$.

Stelling 1 : De GA, toegepast op een matroïde $M = (E,F)$ en een reële functie c op E, construeert een basis van M met maximaal gewicht.

Bewijs : $B = B_n$ is een basis van M wegens lemma 2. Laten B en een willekeurige basis B' van M geordend zijn naar niet-stijgende waarde van hun elementen.
We veronderstellen: er is een element van B, zeg e_k, met een kleiner gewicht dan het element op de overeenkomstige plaats van B', zeg e_j (dus $j < k$).
O.g.v. lemma 2 is $r(A_j) = |B_j| = r(B_j)$. Omdat $B_k \neq B_j$ volgt $r(A_j) < |B_k|$. Maar $B_j' = \{e_1 \mid e_1 \in B', 1 \leq j\}$ is een onafhankelijke deelvz. van A_j met $|B_k|$ elementen, zodat $r(A_j) \geq |B_k|$.
De veronderstelling leidt tot een tegenspraak. Ieder element van B' heeft een gewicht dat niet groter is dan het gewicht van het overeenkomstige element van B, zodat $c(B') \leq c(B)$. q.e.d.

1.3. Toepassingen.

1.3.1. Als we niet een basis van de matroïde $M = (E,F)$ zoeken, maar een onafhankelijke vz. met maximaal gewicht, kunnen we als volgt te werk gaan.
E^+ is de deelvz. van E van elementen met positief gewicht. We definiëren op E^+ de matroïde $M^+ = (E^+, F \cap E^+)$, de *restrictie* van F tot E^+, met als onafhankelijke vzn. de onafhankelijke vzn. van M die bevat zijn in E^+. Het is duidelijk dat M^+ aan definitie 1 voldoet.
De GA toegepast op M^+ construeert een basis B^+ van M^+ met maximaal gewicht. B^+ is de gezochte vz., omdat elementen uit $E \setminus E^+$ niet in een maximaal wegende deelvz. bevat kunnen zijn, en onafhankelijke deelvz. van M^+, die geen basis zijn van M^+, uitgebreid kunnen worden tot een basis van M^+. We krijgen hetzelfde resultaat door de GA toe te passen op M, waarbij in stap 1 B_j slechts

dan gelijk wordt gesteld aan $B_{j-1} \cup \{e_j\}$ als $B_{j-1} \cup \{e_j\} \in F$ en $c_j > 0$.

1.3.2. Een bedrijf heeft n orders in portefeuille. Van iedere order is bekend op welke van de beschikbare m machines hij kan worden uitgevoerd. Iedere order vergt één week produktie op de machine waarop hij is ingedeeld. Het is het bedrijf c_j waard, als order j in de komende week wordt uitgevoerd. Gevraagd wordt een optimaal produktieschema voor de komende week te ontwerpen. Het bedrijf verstaat daaronder een indeling, waarbij allereerst zoveel mogelijk orders worden uitgevoerd, welke orders bovendien een zo groot mogelijke gezamenlijke waarde moeten hebben.

Zij $E = \{1,\ldots,n\}$ de vz. van orders. Een deelvz. orders is onafhankelijk als de orders uit deze deelvz. de komende week allen uitgevoerd kunnen worden. Mits het aldus gedefinieerde paar $M = (E,F)$ een matroïde is, geeft de GA een basis B van maximaal gewicht, die juist aan de optimaliteitseisen voldoet.
In stap 1 van de GA moeten we bepalen of $B_{j-1} \cup \{j\}$ een onafhankelijke vz. is. Definiëren we probleem P(B) als

$$\max \sum_{i=1}^{m} \sum_{j \in B} a_{ij} x_{ij}$$

$$\sum_{j \in B} x_{ij} \leq 1 \quad , \quad i = 1,\ldots,m$$

$$\sum_{i=1}^{m} x_{ij} \leq 1 \quad , \quad j \in B$$

$$x_{ij} \in \{0,1\} \quad , \quad \text{voor alle i en j ,}$$

$$\text{met} \quad a_{ij} = \begin{cases} 1 & , \text{ order j is uitvoerbaar op machine i} \\ 0 & , \text{ anders,} \end{cases}$$

dan moeten we in feite nagaan, of $P(B_{j-1} \cup \{j\})$ een optimale oplossing met waarde $|B_{j-1}| + 1$ heeft. Een efficiënte methode hiervoor geeft b.v. [15], p. 55 e.v.
De optimale indeling wordt gevonden door oplossing van probleem $P(B_n)$.
Alle orders kunnen slechts dan in de komende week uitgevoerd worden, als

$r(E) = |E|$.

De matroïde M = (E,F) is een zogenaamde *transversaalmatroïde*.
Zij E een eindige vz. en $J = (S_1,\ldots,S)$ een familie van niet-lege deelvzn. van E. Een vz. $A \subseteq E$ heet een transversaal (system of distinct representatives) van J als A uit m elementen bestaat, één uit iedere vz. S_i.
A is een partiële transversaal als A transversaal is van een deelfamilie van J. Een matroïde M= (E,F), waarvan de onafhankelijke vzn. de partiële transversalen zijn van een familie $J = (S_1,\ldots,S_m)$, met $S_i \subseteq E$, heet een transversaalmatroïde [25].
We moeten nog aantonen dat op deze manier inderdaad een matroïde is verkregen. Een deelvz. van een partiële transversaal is opnieuw een partiële transversaal, dus de partiële transversalen vormen een onafhankelijk stelsel. Als bewezen is dat dit stelsel aan eigenschap (3) voldoet, is M = (E,F) een matroïde.
I en J zijn twee partiële transversalen van J met k, resp. k + 1 elementen. Element $e \in I$ vertegenwoordigt vz. $\psi(e)$ en I vertegenwoordigt $\psi(I)$. Evenzo vertegenwoordigt $e \in J$ $\chi(e)$ en J vertegenwoordigt $\chi(J)$. $\chi(J)$ heeft een element meer dan $\psi(I)$, zodat er een element in J is, zeg e_1, zó dat $\chi(e_1) \notin \psi(I)$.
We onderscheiden twee situaties:

a) $e_1 \notin I$: $I \cup \{e_1\}$ is een transversaal van $\psi(I) \cup \{\chi(e_1)\}$ en we zijn klaar ;

b) $e_1 \in I \cap J$. We herdefiniëren nu $\psi(e_1) = \chi(e_1)$.

Omdat blijft gelden $|\chi(J)| = |\psi(I)| + 1$, kunnen we deze stap herhalen voor elementen $e_2, e_3, \ldots\ldots$. Steeds is $e_k \notin \{e_1,\ldots,e_{k-1}\}$, omdat na herdefinitie geldt $\chi(\{e_1,\ldots,e_{k-1}\}) = \psi(\{e_1,\ldots,e_{k-1}\})$. Bij voortzetting van de iteratie treedt situatie a) op, of na ten hoogste $|I \cap J|$ stappen is $\chi(I \cap J) = \psi(I \cap J)$. Een element e met $\chi(e) \notin \psi(I)$ kan dan niet tot $I \cap J$ behoren: $e \in J \setminus I$ en $I \cup \{e\}$ is een transversaal van $\psi(I) \cup \{\chi(e)\}$.

1.3.3. Het bedrijf uit toepassing 1.3.2. kent geen waarde c_j aan de orders toe, maar rangschikt deze naar dalende prioriteit; order k heeft prioriteit boven order j als k < j, voor alle k en j. Bestaat er zoiets als een optimaal produktieschema [18]?
Produktieschema's die een verschillend aantal orders indelen, zijn moeilijk

vergelijkbaar. Echter, onafhankelijk van het gehanteerde kriterium, is de vz. orders in een realiseerbaar produktieschema een onafhankelijke vz. van de matroïde M = (E,F), op dezelfde manier gedefinieerd als boven. Het maximum aantal in één week uit te voeren orders is r(E), en elk produktieschema dat uit minder orders bestaat, kan worden uitgebreid tot een schema van r(E) orders. We behoeven dus alleen schema's, afkomstig van bases van M, te vergelijken.
De oplossing die door de GA wordt geconstrueerd bij een gegeven matroïde M = (E,F), is alleen afhankelijk van de rangschikking van de elementen van E. Voor de orders gerangschikt naar dalende prioriteit, construeert de GA een lexicografisch maximale oplossing $B = \{e_1,\ldots,e_{r(E)}\}$. Dezelfde basis vinden we als we de GA toepassen met een rangschikking naar dalende gewichten $c_j = n - j$ i.p.v. naar prioriteiten. Als $B' = \{e'_1,\ldots,e'_{r(E)}\}$ een andere basis is van M, en B en B' zijn beiden gerangschikt naar dalende gewichten, dan volgt uit het bewijs van stelling 1 :

$$c_{e_j} \geq c_{e'_j} \quad , \quad \text{voor } j = 1,\ldots,r(E) \ ,$$

m.a.w. iedere order in B, die niet gelijk is aan de overeenkomstige order in B', heeft een hoger gewicht, en derhalve een hogere prioriteit dan deze laatste order. Het lexicografisch maximum is dus met reden optimaal te noemen [26].

Opmerking.
We zien dat de basis B_n, voortgebracht door de GA, ook de waarde (c.q. prioriteit) van het minimale element in de oplossing maximaliseert: B_n is ook een oplossing van het *bottleneckprobleem* [13]:

$$\max_{B \in \mathcal{B}} \ \min_{j \in B} \ c_j \quad ,$$

als $\mathcal{B}$ de vz. is van alle bases van de matroïde M = (E,F).

1.4. Definities van matroïde.

In definitie 1 wordt een matroïde gedefinieerd op grond van een collectie onafhankelijke vzn. Daarnaast zijn definities mogelijk uitgaande van bases, c.q. circuits, c.q. de rangfunctie r [27, 29].
In de volgende definities is E een eindige, niet-lege vz.

Definitie 2. Het paar $M = (E,\mathcal{B})$, met $\mathcal{B}$ een niet-lege collectie deelvzn. van E, genaamd bases, is een matroïde, als geldt:

(4) geen basis is echt bevat in een andere basis;

(5) als B_1 en B_2 bases zijn, bestaat er bij een element $e_1 \in B_1$ een element $e_2 \in B_2$, zó dat $(B_1 \setminus \{e_1\}) \cup \{e_2\}$ ook een basis is (uitwisseleigenschap).

Definitie 3. Het paar $M = (E,\mathcal{C})$, met $\mathcal{C}$ een niet-lege collectie deelvzn. van E, genaamd circuits, is een matroïde, als geldt:

(6) geen circuit is echt bevat in een ander circuit;

(7) als C_1 en C_2 circuits zijn, en $e \in C_1 \cap C_2$, bestaat een circuit in $C_1 \cup C_2$ waarvan e geen deel uitmaakt.

Definitie 4. Het paar $M = (E,r)$, met r een geheelwaardige functie, gedefinieerd op de vz. van deelvzn. van E, is een matroïde, als r voldoet aan:

(8) $0 \leq r(A) \leq |A|$, voor elke $A \subseteq E$;

(9) $r(A) \leq r(B)$, als $A \subseteq B \subseteq E$;

(10) $r(A \cup B) + r(A \cap B) \leq r(A) + r(B)$,
, voor elke $A \subseteq E$ en $B \subseteq E$;

De definities 1, 2, 3 en 4 zijn equivalent. We bewijzen de equivalentie van 1 en 4; de overige bewijzen worden aan de lezer overgelaten.

Stel $M = (E,F)$ is een matroïde gedefinieerd door onafhankelijke vzn.
Voor de functie $r(A)$, $A \subseteq E$, nemen we het aantal elementen van een maximaal onafhankelijke deelvz. bevat in A. De aldus gedefinieerde r voldoet aan (8) en (9).
Zij X een maximaal onafhankelijke vz. van $A \cap B$, voor $A \subseteq E$ en $B \subseteq E$.

X kan worden uitgebreid tot een maximaal onafhankelijke vz. van $A \cup B$, zeg Z. $Z \cap (A \quad B) = X$, anders zou X niet maximaal zijn voor $A \cap B$. $Z \cap A$ is een onafhankelijke deelvz. van A, evenals $Z \cap B$ van B.
We zien:

$$\begin{aligned} r(A \cup B) = |Z| &= |Z \cap A| + |Z \cap B| - |X| \\ &\leq r(A) + r(B) - r(A \cap B). \end{aligned}$$ q.e.d.

Omgekeerd, stel $M = (E,r)$ is een matroide, gedefinieerd door de rangfunctie r. We noemen een vz. $A \subseteq E$ onafhankelijk als $r(A) = |A|$. Als A een deelvz. is van een onafhankelijke vz. B, volgt uit (10) en (8):

$$|B| = r(B) \leq r(A) + r(B \setminus A) \leq r(A) + |B| - |A|.$$

Wegens (8) geldt $r(A) \leq |A|$, zodat $r(A) = |A|$, m.a.w.: A is onafhankelijk.
We moeten nog aantonen dat twee maximaal onafhankelijke deelvzn. B_1 en B_2 van een vz. $A \subseteq E$ evenveel elementen hebben. Zonder verlies van algemeenheid stellen we $|B_1| \leq |B_2|$. Voor elke $e \in B_2 \setminus B_1$ geldt $r(B_1 \cup \{e\}) = |B_1|$, anders is B_1 niet maximaal.
O.g.v. (10) vinden we voor $e \neq f$, $\{e,f\} \subseteq B_2 \setminus B_1$:

$$\begin{aligned} r(B_1 \cup \{e\} \cup \{f\}) &\leq r(B_1 \cup \{e\}) + r(B_1 \cup \{f\} - r(B_1) \\ &= |B_1|. \end{aligned}$$

Derhalve is $r(B_1 \cup \{e\} \cup \{f\}) = |B_1|$. Voegen we de elementen van $B_2 \setminus B_1$ één voor één aan B_1 toe, dan vinden we uiteindelijk $r(B_1 \cup B_2) = |B_1|$.
Uit de onafhankelijkheid van B_2 en uit (9) volgt:

$$|B_2| = r(B_2) \leq r(B_1 \cup B_2) = |B_1|.$$

Derhalve geldt $|B_2| = |B_1|$. q.e.d.

2. Polyhedra.

2.1. Polyhedra van matroïden.

Een onafhankelijke vz. I van de matroïde M = (E,F), waarbij E de vz. {1,...,n} is, kan worden gekarakteriseerd door een 0,1-incidentievector van n componenten $x = (x_1,\ldots,x_n)$:

$$x_j = \begin{cases} 1 & , \ j \in I \\ 0 & , \ \text{anders.} \end{cases}$$

Als V, resp. V_B, de vz. is van alle incidentievectoren van onafhankelijke vzn., resp. bases, van M, en $c = (c_1,\ldots,c_n)$ is een reële vector, dan bepaalt de GA een oplossing van het probleem $\max\{cx \mid x \in V_B\}$. Dit probleem is een lineair programmeringsprobleem, nl. maximalizeer cx over de hoekpunten van het polyhedron P_B, gedefinieerd als de convex omhullende van V_B. Evenzo is het probleem $\max\{cx \mid x \in V\}$ equivalent aan het maximaliseren van cx over de hoekpunten van P, de convex omhullende van V.
De polyhedra P en P_B, die gegenereerd worden door de matroïde M met rangfunctie r, zijn eenvoudig te beschrijven door lineaire stelsels L resp. L_B. L is het lineaire stelsel [(11) - (12)], L_B het stelsel [(11) - (12) - (13)]:

$$(11) \qquad \sum_{j\in A} x_j \leq r(A) \quad , \quad \text{voor alle} \quad A \subseteq E.$$

$$(12) \qquad x_j \geq 0 \quad , \quad \text{voor alle} \quad j \in E.$$

$$(13) \qquad \sum_{j\in E} x_j = r(E) \quad .$$

Met H, resp. H_B, geven we de vz. hoekpunten van polyhedron P, resp. P_B, gedefinieerd door L, resp. L_B, aan. In stelling 2 wordt bewezen H = V.
Het bewijs van Edmonds [11] maakt gebruik van de eigenschappen van de GA en laat en passant opnieuw zien dat de GA een optimale onafhankelijke vz. van de matroïde M construeert. Een soortgelijke bewijsvoering wordt door Edmonds gehanteerd om de juistheid van twee andere algoritmen te bewijzen, nl. de algoritme die een optimale branching construeert [8](zie § 3), en de algo-

ritme ter bepaling van een maximale (gewogen) koppeling [5, 6]. Schematisch weergegeven verloopt het bewijs als volgt:

0. V is de vz. oplossingen van het beschouwde probleem;
   P is een polyhedron, beschreven door een lineair stelsel L;
   H is de vz. hoekpunten van P.
1. Er is een - al dan niet efficiënte - algoritme, die een element v^o van V construeert.
2. $V \subseteq H$, omdat iedere $v \in V$ aan L voldoet en voorgesteld kan worden als een basisoplossing van L, i.e. de unieke oplossing van een aantal gelijkheden, die verkregen zijn uit ongelijkheden van L door het $\leq$-teken te vervangen door een gelijkteken.
3. Voor een willekeurige lineaire functie cx wordt het max $\{cx|x \in P\}$ aangenomen door een oplossing in V, en wel de v^o die geconstrueerd wordt door de algoritme uit punt 1. Het bewijs van deze stap is gebaseerd op de dualiteitsstelling uit de lineaire programmering.
4. Een hoekpunt h van P is een punt van P, dat een lineaire functie, zeg $c_h x$, uniek maximaliseert over P. Volgens 3. is deze unieke h een element van V, m.a.w. $H \subseteq V$.
5. $H = V$ o.g.v. 2. en 4.
6. v^o is een optimale oplossing van het beschouwde probleem voor een kriteriumfunctie cx o.g.v. 3. en 5..
7. Uit 1. en 6. volgt dat er een (efficiënte) algoritme bestaat ter bepaling van een optimale oplossing van het beschouwde probleem.

Stelling 2. De vz. hoekpunten H van polyhedron P, gedefinieerd door stelsel L = [(11) - (12)], is precies de vz. V van incidentievectoren van onafhankelijke vzn. van de matroïde M.

Bewijs : De bewijsgang van Edmonds volgend, moeten we de punten 1, 2 en 3 waarmaken.

ad 1. We passen de GA toe op de matroïde $M^+ = (E^+, F \cap E^+)$.
   $E^+ = \{1,\ldots,m\} \subseteq E$ is de vz. elementen van E met positief gewicht, zo genummerd dat $c_1 \geq c_2 \geq \ldots \geq c_m > 0$.
   De GA construeert een onafhankelijke vz. $B \subseteq E^+$((§ 1.3.1.).

Als v^o nemen we de incidentievector van B, dwz. v^o_j is 1 als $j \in B$ en 0 als $j \in E \setminus B$.
Voor de deelvzn. $A_j = \{1,\ldots,j\}$ en B_j, die optreden bij toepassing van de GA, geldt volgens lemma 2 :

$$r(A_j) = |B_j| \quad , \text{ voor } 0 \leq j \leq m.$$

Als $j \in B$, dan $r(A_j) = |B_j| = |B_{j-1} \cup \{j\}| = |B_{j-1}| + 1 = r(A_{j-1}) + 1$.
Als $j \in E^+ \setminus B$, dan $r(A_j) = |B_{j-1}| = r(A_{j-1})$. We kunnen de incidentievector v^o derhalve definiëren als

$$v^o_j = \begin{cases} r(A_j) - r(A_{j-1}) & , \ j \in E^+ \\ 0 & , \ j \in E \setminus E^+ \end{cases} ,$$

ad 2. Een incidentievector x van een onafhankelijke vz. I voldoet aan (11) omdat voor elke $A \subseteq E$ de vz. $A \cap I$ een onafhankelijke deelvz. van A is en de waarde van het linkerlid van (11) is $|A \cap I| \leq r(A)$.
De 0,1-vector x is dus bevat in P. x is een hoekpunt van P, zijnde snijpunt van de n hypervlakken $x_j = 0$ voor $j \in E \setminus I$ en $x_j = r(\{j\})$ voor $j \in I$.

ad 3. We moeten bewijzen dat voor $x = v^o$ het maximum van het l.p.-probleem $\max \{cx \mid x \in P\}$ wordt aangenomen. Het duale probleem luidt:

$$\min \sum_{A \subseteq E} r(A) u_A$$

onder:

$$\sum_{A \ni j} u_A \geq c_j \quad , \text{ voor alle } j \in E .$$

$$u_A \geq 0 \quad , \text{ voor alle } A \subseteq E .$$

We definieren de vector u^o als:

$$u^o_A = \begin{cases} c_j - c_{j+1} & , \ A = A_j \text{ voor } 1 \leq j \leq m-1, \\ c_m & , \ A = A_m \ , \\ 0 & , \ \text{anders.} \end{cases}$$

u^o voldoet aan de bijvoorwaarden van het duale probleem en voorts is

$$\sum_{A \subseteq E} r(A)u_A^o = \sum_{j=1}^{m-1} r(A)\{c_j - c_{j+1}\} + r(A_m)\, c_m$$

$$= \sum_{j=1}^{m} c_j\{r(A_j) - r(A_{j-1})\} = \sum_{j=1}^{m} c_j v_j^o \,.$$

Volgens de dualiteitstelling uit de lineaire programmering vormen v^o en u^o een optimaal paar oplossingen van het primale en het duale probleem.

Gevolg: De vz. hoekpunten H_B van het polyhedron P_B, gedefinieerd door stelsel L_B, is precies de vz. V_B van incidentievectoren van bases van de matroïde M. Dit vloeit voort uit het feit dat een basis van M een hoekpunt is van P, dat bovendien voldoet aan (13).

$H \subseteq V$ kan ook rechtstreeks worden bewezen door aan te tonen dat alle hoekpunten van P geheelwaardig zijn, dwz. de matrix van stelsel L is unimodulair. Als n.l. hoekpunt h van P geheelwaardig is, dan is h een 0,1-vector en is h incidentievector van de vz. $I \equiv \{j \mid j \in E, h_j = 1\}$. Omdat $h \in P$, geldt $\sum_{j \in I} h_j \leq r(I)$. Het aantal elementen van I is gelijk aan $\sum_{j=1}^{n} h_j = \sum_{j \in I} h_j$, zodat $|I| \leq r(I)$: I is een onafhankelijke vz. van de matroïde M. Omgekeerd kan de bewijsvoering van Edmonds worden gebruikt om aan te tonen dat de hoekpunten van een polyhedron geheelwaardig zijn.

2.2. Maximale koppelingen.

In een graaf $G = (V, E)$ heet een vz. kanten $M \subseteq E$ een *koppeling* als ieder punt van V incident is met ten hoogste één kant van de koppeling. Een *b-koppeling* is een vz. kanten $M \subseteq E$, zodat ieder punt $i \in V$ incident is met ten hoogste b_i kanten van M. Een maximale koppeling is een koppeling met een maximaal aantal kanten; een maximaal gewogen koppeling is een koppeling met een maximaal gewicht van de kanten. Een maximaal gewogen b-koppeling is de oplossing van het geheeltallig l.p.-probleem:

$$\max \quad cx$$

onder:

$$\begin{cases} I_G x \leq b \\ 0 \leq x \leq 1 \end{cases} \tag{14}$$

$$x \text{ geheel}, \tag{15}$$

met I_G de punt-kant-incidentiematrix van G.

Als G bipartite is, is I_G unimodulair; de hoekpunten van het polyhedron $\{x \mid I_G x \leq b, 0 \leq x \leq 1\}$ zijn dan precies de incidentievectoren van koppelingen van G. Het maximaal gewogen koppelingsprobleem is voor een bipartite graaf equivalent aan het LAP.
Voor een willekeurige graaf G behoeft I_G niet unimodulair te zijn. Een niet-bipartite graaf G verschilt van een bipartite graaf, doordat in G één of meer cycles met een oneven aantal punten voorkomen. Als G bestaat uit één oneven cycle C met $2k + 1$ punten en kanten $1,2,\ldots,2k+1$, voldoet de oplossing $x_1 = x_2 = \ldots = x_{2k+1} = 1/2$ aan de voorwaarden (14).
In een koppeling kunnen echter maximaal k kanten van C optreden, m.a.w. elke koppeling voldoet aan de voorwaarde

$$\sum_{j=1}^{2k+1} x_j \leq k \ .$$

We definieren bij de graaf G = (V,E) het polyhedron P door het volgende stelsel:

$$I_G x \leq 1$$

$$\sum_{j \in S} x_j \leq \tfrac{1}{2}(|S| - 1), \text{ voor } S \subseteq V, |S| > 1 \text{ en oneven},$$

$$x \geq 0 \ ,$$

waarbij $j \in S$ betekent dat kant j beide uiteinden in vz. S heeft.
Balinski [2] bewijst dat de hoekpunten van P precies de incidentievectoren van koppelingen van G zijn. Het koppelingsprobleem is hiermee één der weinige geheeltallige l.p.-problemen waarvan de convex omhullende van de oplos-

singen expliciet bepaald is. Edmonds [6] identificeert de convex omhullende volgens de in de vorige paragraaf geschetste methode. De in stap 1 van het bewijs benodigde algoritme is een efficiënte algoritme om een maximale koppeling, c.q. maximaal gewogen koppeling te bepalen.
ALGOL-procedures van deze algoritmen zijn opgenomen in de Appendix.

Een soortgelijke, eveneens efficiënte algoritme [14] bepaalt een oplossing van het koppelingsprobleem in zijn meest algemene vorm, het geheeltallige l.p.-probleem op grafen:

$$\begin{aligned} \max \quad & \sum_{j\in E} c_j x_j \\ \text{onder:} \quad & \sum_{j\in E} a_{ij}x_j \leq b_i \quad , \quad i \in V \ , \\ & 0 \leq x_j \leq a_j \quad , \quad j \in E \ , \\ & x_j \text{ geheel} \ , \quad j \in E \ , \end{aligned}$$

waarbij a_{ij}, b_i en a_j niet-negatieve gehele getallen zijn, en

$$\sum_{j\in E} |a_{ij}| \leq 2 \ , \ i \in V \ .$$

(a_{ij}) is de punt-kant-incidentiematrix van een graaf $G = (V,E)$.
Een kant $j \in E$ heeft één of twee uiteinden. Ieder uiteinde is of een kop (head) of een staart (tail) en is incident met één punt $i \in V$. Een kant met twee uiteinden is een schakel (link) resp. lus (loop) als deze uiteinden incident zijn met verschillende punten, resp. hetzelfde punt. Een kant is gericht als hij één kop en één staart heeft, anders heet hij niet-gericht.
De j^{de} kolom van (a_{ij}) geeft de aard van kant j aan: kant j heeft twee staarten, één staart, geen uiteinde, één kop of twee koppen incident met punt i, als resp. $a_{ij} = +2, +1, 0, -1$ of -2.
De bijvoorwaarden eisen dat in een oplossing van het probleem kant j ten hoogste multipliciteit a_j heeft, en voorts dat het aantal koppen minus het aantal staarten van de oplossing dat incident is met punt i, ten hoogste b_i is, waarbij dan ieder uiteinde meetelt voor de multipliciteit van de kant waartoe het behoort.

We stippen enkele toepassingen aan.
Een overdekking van punten door kanten in een graaf G = (V,E) is een vz. kanten, zodat ieder punt van V incident is met tenminste één kant van deze vz. Een *minimale overdekking*, i.e. een overdekking met een minimaal aantal kanten is te construeren uit een maximale koppeling M, door in ieder punt $v \in V$, dat niet incident is met een kant van M, een kant incident met v aan M toe te voegen. Omgekeerd wordt uit een minimale overdekking een maximale koppeling gevormd door in punten die incident zijn met meer dan één kant van de overdekking alle kanten op één na uit de overdekking weg te laten [19].
In het *symmetrische handelsreizigersprobleem* (TSP), i.e. het TSP met een symmetrische kostenmatrix (c_{ij}), wordt gezocht naar een Hamiltoncircuit van minimaal gewicht. Een Hamiltoncircuit kan worden gedefinieerd als een samenhangende 2-koppeling van maximale cardinaliteit.
Het LAP bepaalt, als subroutine voor het TSP met n steden, een 2-koppeling van maximale cardinaliteit in een graaf met n punten en kanten (i,j) en (j,i) tussen elk tweetal punten. De resulterende 2-koppeling kan cycles bestaande uit 2 kanten bevatten. In een 2-koppeling van maximale cardinaliteit, berekend op de volledige graaf K_n, dwz. tussen ieder tweetal punten loopt precies één kant, bestaat een cycle minimaal uit 3 kanten. Deze laatste methode geeft dus scherpere ondergrenzen voor het symmetrische TSP dan de oplossing van het LAP en heeft als tweede, wellicht nog belangrijker voordeel, dat in een branch-and-bound aanpak van het TSP geen cycles van 2 kanten geëlimineerd behoeven te worden [3].
Het volgende probleem: "Gegeven n taken die ieder één tijdseenheid duren, een partiële ordening van de taken, geïnduceerd door een acyclische gerichte graaf A, en 2 identieke machines, bepaal een volgorde van de taken, zo dat alle taken zó vroeg mogelijk uitgevoerd zijn", kan worden opgelost met gebruik van de algoritme voor een maximale koppeling [16]. Uit A wordt de compatibiliteits-graaf G = (V,E) afgeleid: V is de vz. taken, een kant $(i,j) \in V$, als taak i tegelijk met taak j kan worden uitgevoerd, dwz. in graaf A bestaat geen gericht pad van i naar j, noch van j naar i. In G wordt een maximale koppeling bepaald. Als deze koppeling m kanten heeft, $0 \leq m \leq [n/2]$, zal iedere indeling van de taken tenminste n-m tijdseenheden vergen. Uit de berekende maximale koppeling en de gegeven partiële ordening kan op eenvoudige wijze een indeling worden afgeleid, die n-m tijdseenheden vergt en dus optimaal is.

3. Een algoritme voor een optimale branching.

3.1. Inleiding.

We voeren allereerst een aantal begrippen in.
G = (V,E) is een gerichte graaf. Een *cycle* Q is een samenhangende graaf Q, zó, dat ieder punt van Q incident is met precies twee kanten van Q. Een *circuit* is een cycle, zó, dat naar ieder punt van het circuit precies één kant gericht is. Een *bos (forest)* is een graaf zonder cycles. Een *boom (tree)* is een samenhangend bos. Een *vertakking (branching)* is een bos, waarvan geen twee kanten naar hetzelfde punt gericht zijn. Een *arborescence* is een samenhangende branching. Een opspannende arborescence van G is een arborescence die incident is met alle punten van G. Een *pad* P is een arborescence zodat iedere kant in P vertrekt van een ander punt in P. Een *Hamiltonpad* in G is een pad dat incident is met alle punten van G. Een *wortel* van een vz. S van gerichte kanten, is een punt, incident met S, zó, dat geen kant van S gericht is naar de wortel.

Kant $e_k \in E$ heeft een gewicht c_k. Het gewicht van een vz. kanten is de som van de gewichten van de kanten in de vz.. Een branching heet *optimaal* als zijn gewicht maximaal is onder alle branchings op G.
Een Hamiltonpad in G van punt x naar punt y is een opspannende arborescence met wortel x, die bovendien de eigenschap heeft dat van ieder punt ten hoogste één kant weggaat. Berekeningen van een minimale arborescence geeft een ondergrens voor het minimale Hamiltonpad in G en is bruikbaar als subroutine in een branch-and-bound aanpak van het asymmetrische TSP [21].
Een praktisch voorbeeld van een arborescence is het volgende.
c_k zijn de kosten om informatie over te brengen van het begin van kant e_k naar het eind. Als de informatiekosten additief zijn, geeft een minimale opspannende arborescence met wortel w de goedkoopste manier om alle punten vanuit w te informeren.

3.2. Matroïde formulering.

We karakteriseren een branching als een vz. kanten die onafhankelijk is zowel in een matroïde $M_1 = (E,F_1)$ als in een matroïde $M_2 = (E,F_2)$. Het bepalen

van een optimale branching is het bepalen van een vz. met maximaal gewicht in $F_1 \cap F_2$.
Zij F_1 de familie van alle vzn. kanten van G met de eigenschap dat geen twee kanten van de vz. naar hetzelfde punt G gericht zijn. Het stelsel $M_1 = (E,F_1)$ is een matroide: M_1 is de transversaalmatroide die voortgebracht wordt door de familie $S = (S_1,\ldots S_{|V|})$, met S_i de vz. kanten gericht naar punt i. We kunnen ook rechtstreeks bewijzen dat M_1 voldoet aan de definitie van matroiden. Eigenschap(1) is triviaal. Eigenschap (2) eist dat iedere maximaal onafhankelijke deelvz. van een vz. $S \subseteq E$ evenveel elementen heeft. Als V_S de vz. punten van G is, waarnaar een kant van S gericht is, dan is een vz. $B \subseteq S$ maximaal onafhankelijk in S d.e.s.d. als naar ieder punt van V_S precies één kant van B gericht is. Elk maximaal onafhankelijke vz. B van S heeft $|V_S|$ elementen.
Zij $M_2 = (E,F_2)$ de graafmatroïde bij G, dwz. F_2 is de familie van alle bossen in G.
Het is duidelijk dat een vz. in de doorsnede van de aldus gedefinieerde F_1 en F_2 een branching is, en omgekeerd.

We hebben in § 2 gezien dat we bij een matroïde M_i een polyhedron P_i kunnen aangeven, en wel zo, dat de vz. hoekpunten $H(P_i)$ van P_i gelijk is aan de verzameling I van incidentievectoren van onafhankelijke vzn. van de matroïde M_i. Een andere stelling van Edmonds is:

Stelling 3: $H(P_1 \cap P_2) = H(P_1) \cap H(P_2)$.

Bewijs : Zie [10].

In woorden: de punten die hoekpunten zijn van P_1 en P_2, zijn hoekpunt van de doorsnede van P_1 en P_2, en omgekeerd. De incidentievectoren x van onafhankelijke vzn. in M_1 en M_2 zijn derhalve hoekpunten van de doorsnede van P_1 en P_2. Als L_1 en L_2 lineaire stelsels zijn die P_1 resp. P_2 bepalen, is het probleem: zoek een maximale vz. in $F_1 \cap F_2$ equivalent met: maximaliseer cx over de hoekpunten van $L_1 \cup L_2$.
Voor het algemene probleem, waarbij M_1 en M_2 willekeurige matroïden zijn, is een efficiënte algoritme ontwikkeld door Edmonds en Lawler [11, 24].

Voor een optimale branching zoeken we de doorsnede van twee speciale matroïden, nl. een graafmatroïde en een transversaalmatroïde. De stelsels L_1 en L_2 kunnen in dit geval aanzienlijk gereduceerd worden. De gereduceerde stelsels kunnen ook rechtstreeks worden afgeleid. De benadering via matroïden verschaft de wetenschap dat de hoekpunten van de stelsels geheeltallig zijn.
We kunnen de problemen, waarvan de oplossingen gemeenschappelijke onafhankelijke vzn. van één of meer matroïden zijn, classificeren naar het aantal matroïden dat minimaal nodig is om een oplossing van het probleem te verkrijgen. Een probleem heet een p-matroïde probleem als dit aantal p is.
Het STP is een 1-matroïde probleem (§ 1.1). We hebben boven gezien dat het bepalen van een optimale branching een 2-matroïde probleem is.
Het lineaire transportprobleem is eveneens een 2-matroïde probleem. In de bipartite graaf met puntvzn. X en Y is bij ieder punt $x \in X$, resp. $y \in Y$ een geheel getal a_x, resp. b_y gegeven, zodat geldt: $\sum_{x\in X} a_x = \sum_{y\in Y} b_y$.
Tussen punten x en y lopen min $\{a_x, b_y\}$ kanten, ieder met gewicht c_{xy}.
De matroïde $M_1 = (E,F_1)$, met E de vz. kanten van de bipartite graaf, heeft als onafhankelijke vzn. de vzn. $I \subseteq E$, waarvan ten hoogste a_x kanten incident zijn met ieder punt $x \in X$. In $M_2 = (E,F_2)$ is een vz. onafhankelijk als ten hoogste b_y kanten van de vz. incident zijn met ieder punt $y \in Y$. Het LAP is een lineair transportprobleem waarin alle a_x en b_y één zijn. Het is geen 1-matroïdeprobleem (§ 1.1), zodat het LAP en het lineaire transportprobleem 2-matroïdeproblemen zijn.
Voor 1- en 2-matroïdeproblemen bestaan efficiënte algoritmen, voor p-matroïdeproblemen, $p \geq 3$, niet. Bij het zoeken naar een efficiënte algoritme voor $p = 3$ stuiten we al direct op de moeilijkheid dat het equivalent van stelling 3 voor $p = 3$:

$$H(P_1 \cap P_2 \cap P_3) = H(P_1) \cap H(P_2) \cap H(P_3)$$

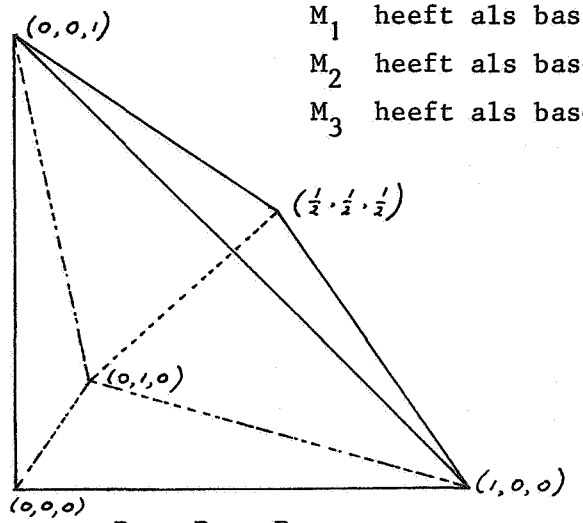
i.h.a. niet geldt (zie voorbeeld 3).

Voorbeeld 3. $E = \{1,2,3\}$.

M_1 heeft als bases $\{1,2\}$ en $\{2,3\}$,

M_2 heeft als bases $\{1,3\}$ en $\{2,3\}$,

M_3 heeft als bases $\{1,2\}$ en $\{1,3\}$.



$P_1 \cap P_2 \cap P_3$

$(\frac{1}{2},\frac{1}{2},\frac{1}{2})$ is hoekpunt van $P_1 \cap P_2 \cap P_3$, maar niet van P_i, $i = 1,2,3$.

Het handelsreizigersprobleem (TSP) is een 3-matroïde probleem, en als zodanig voor 2/3 opgelost, aldus Edmonds [11]. De matroïden $M_i = (E,F_i)$, $i = 1,2,3$ voor het TSP hebben de volgende onafhankelijke vzn. $I \subseteq E$, met E de vz. kanten van de volledige gerichte graaf $G = (V,E)$:

1. $I \in F_1$, als geen twee kanten van I naar hetzelfde punt van G gericht zijn;
2. $I \in F_2$, als geen twee kanten van I van hetzelfde punt van G vertrekken;
3. $I \in F_3$, als I een 1-boom is, dwz. de kanten van I die niet incident zijn met punt 1, bevatten geen cycle, en met punt 1 zijn hoogstens twee kanten incident.

Als $r(M_i) = |V|$, $i = 1,2,3$, zijn de gemeenschappelijke bases van M_1, M_2 en M_3 juist de Hamiltoncircuits van G. De gemeenschappelijke bases van M_1 en M_2 zijn oplossingen van het LAP in de volledige bipartite graaf met vzn. punten $X = V$ en $Y = V$. De gemeenschappelijke bases van M_1 en M_3 zijn oplossingen van het 1-arborescence probleem (S1AP); een 1-arborescence is een vz. kanten $A \subseteq E$, zó, dat naar ieder punt van G precies 1 kant van A gericht is, en A één cycle door punt 1 heeft. Voor het LAP en het S1AP, 2-matroïde problemen, bestaan goede algoritmen, die gebruikt kunnen worden als subroutines in een branch-and-bound aanpak van het asymmetrische TSP [21].

3.3. L.p.-formulering.

Stelsel L_i, dat polyhedron P_i bij matroide M_i bepaalt, is (zie § 2.1) :

$$(16) \qquad \begin{cases} \sum_{j\in A} x_j \leq r_i(A) , & \text{voor alle } A \subseteq E \\ x_j \geq 0 , & \text{voor alle } j \in E. \end{cases}$$

Voor stelsel L_1 is rang $r_1(A)$ het aantal punten van G, waarnaar kanten van vz. A gericht zijn. Als we voor A alle kanten gericht naar een punt v nemen, is $r_1(A) = 1$.
Stelsel L_1 bevat het volgende stelsel:

$$(17) \qquad \begin{cases} \sum_{j|h_j=v} x_j \leq 1 , & \text{voor alle } v \in V, \\ x_j \geq 0 , & \text{voor alle } j \in E, \end{cases}$$

met h_j het einde van kant $e_j = (t_j,h_j)$.
Omgekeerd impliceert stelsel (17) ook L_1 :

$$\sum_{j\in A} x_j \leq \sum_{v\in V_A} \sum_{j|h_j=v} x_j \leq |V_A| = r_1(A) .$$

L_2 herschrijven we als volgt. Zij $J = \{S|S \subseteq V, |S| \geq 2\}$.
Voor een $S \in J$ geldt:

$$(18) \qquad \sum_{j|\{t_j,h_j\}\subseteq S} x_j \leq r_2(\{e_j|e_j = (t_j,h_j), \{t_j,h_j\}\subseteq S\}) \leq |S| - 1 .$$

Omgekeerd, stel (18) geldt voor alle $S \in J$. A is een vz. kanten van G, bestaande uit p disjuncte vzn. $A_1,\ldots,A_p$, en V_A is de vz. punten incident met A. Dan

$$\sum_{j\in A} x_j = \sum_{i=1}^{p} \sum_{j\in A_i} x_j$$

$$\leq \sum_{i=1}^{p} \sum_{j|\{t_j,h_j\}\subseteq V_{A_i}} x_j$$

$$\leq \sum_{i=1}^{p} (|V_{A_i}|-1)$$

$$= \sum_{i=1}^{p} r_2(A_i)$$

$$= r_2(A)$$

Een optimale branching wordt derhalve gevonden als een hoekpuntoplossing van het l.p.-probleem:

(19) maximaliseer $\sum_{j=1}^{n} c_j x_j$

onder:

(20) $$\sum_{j|h_j=v} x_j \leq 1 \quad , \text{ voor alle } v \in V$$

(21) $$\sum_{j|\{h_j,t_j\}\subseteq S} x_j \leq |S| - 1 \quad , \text{ voor alle } S \in J$$

(22) $$x_j \geq 0 \quad , \text{ voor alle } j \in E$$

Het duale probleem luidt:

(23) minimaliseer $\sum_{v\in V} u_v + \sum_{S\in J} (|S|-1)\, y_S$

onder:

(24) $$u_{h_j} + \sum_{S|S\supseteq\{t_j,h_j\}} y_S \geq c_j \quad , \text{ voor alle } j \in E$$

(25) $$u_v \geq 0 \quad , \text{ voor alle } v \in V$$
$$y_S \geq 0 \quad , \text{ voor alle } S \in J.$$

Een branching B is optimaal d.e.s.d. als we een duale oplossing (u,y) kunnen construeren, zó, dat (u,y) en incidentievector x van B voldoen aan de voorwaarden (20) - (22), (24) - (25) en hun kriteriumwaarden gelijk zijn, dwz.:

(26) $$e_j \in B \Rightarrow u_{h_j} + \sum_{S \mid S \supseteq \{t_j, h_j\}} y_S = c_j$$

(27) $$u_v > 0 \Rightarrow \text{een kant van B is naar v gericht}$$

(28) $$y_S > 0 \Rightarrow \sum_{j \mid \{t_j, h_j\} \subseteq S} x_j = |S| - 1$$

("de branching beperkt tot punten van S heeft $|S| - 1$ kanten")

De algoritme werkt niet expliciet met deze optimaliteitsvoorwaarden. Zij construeert een branching B op een zodanige manier dat bij B altijd een duale oplossing kan worden gevonden met dezelfde kriteriumwaarde als B.

3.4. Algoritme.

L0: Initaliseer:
$G^0 = G$, $V^0 = V$, $E^0 = E$;
$W^0 = V$ (W^i is de vz. van nog niet beschouwde punten in G^i) ;
$S^0 = \emptyset$ (S^i is de vz. van geselecteerde kanten in G^i) ;
$i = 0$.

L1: Als $W^i = \emptyset$, dwz. alle punten van G^i zijn beschouwd, ga dan naar N1.

L2: Kies $v \in W^i$ en stel $W^i := W^i \setminus \{v\}$.

L3: Bepaal een kant in E^i, gericht naar v, met maximaal positief gewicht onder de kanten gericht naar V. Als zo'n kant bestaat, zeg $e_k = (w,v)$, stel dan $S^i := S^i \cup \{e_k\}$.

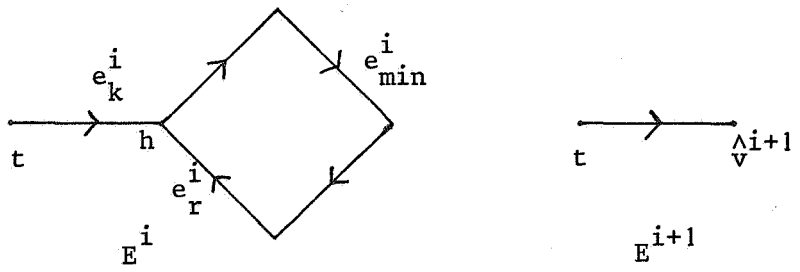
L4: Als S^i een branching is in G^i, dwz. er is in S^i geen pad van v naar w, ga dan naar L1.

M1: Er is in S^i een uniek circuit Q^i, gevormd door e_k en het pad van v naar w in S^i. Genereer een nieuwe graaf G^{i+1} door de punten en kanten in Q^i in te krimpen tot één punt als volgt:

V^{i+1} bevat de punten van V^i die niet incident zijn met Q^i en een punt $\hat{v}^{i+1}$;

E^{i+1} bevat alle kanten van E^i die ten hoogste één uiteinde in Q^i hebben; een kant in E^i met uiteinde in Q^i heeft in E^{i+1} punt $\hat{v}^{i+1}$ als dat uiteinde, de overige uiteinden zijn dezelfde voor E^i en E^{i+1}.

M2: De gewichten c_k^{i+1} zijn onveranderd voor kanten e_k^{i+1} niet gericht naar $\hat{v}^{i+1}$, dwz. $c_k^{i+1} = c_k^i$. Als e_k^{i+1} gericht is naar $\hat{v}^{i+1}$, zeg $e_k^i = (t,h)$ en h is incident met Q^i, dan wordt een correctie toegepast op c_k^i (Fig. 2):



Figuur 2

$$c_k^{i+1} = c_k^i + c_{min}^i - c_r^i ,$$

waarbij e_{min}^i een kant in Q^i is met minimaal gewicht onder de kanten van Q^i, en e_r^i de kant in Q^i gericht naar h.

M3: Stel $W^{i+1} = W^i \cap V^{i+1} \cup \{\hat{v}^{i+1}\}$ en $S^{i+1} = S^i \setminus Q^i$.
S^{i+1} is een branching in G^{i+1}, omdat naar $\hat{v}^{i+1}$ geen kant van S^{i+1} is gericht en voor de overige punten van V^{i+1} geen verschil bestaat t.o.v.

de situatie in G^i.

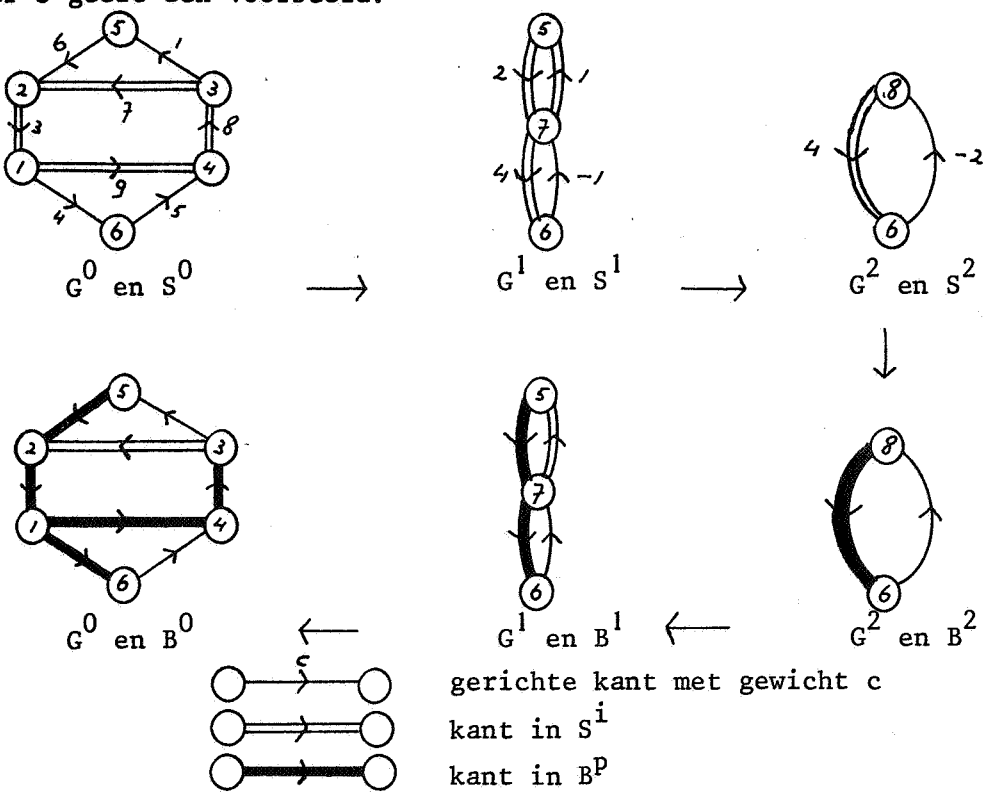
M4: Stel i := i+1. Ga naar L1;

N1: Stel p = i en $B^p = S^p$. B^p is een branching in G^p.
Ga naar N3.

N2: p := p - 1. Bepaal branching B^p en G^p als volgt.
$B^{p+1} \cup Q^p$ bevat precies één circuit, n.l. Q^p. Als in G^p geen kant van B^{p+1} gericht is, verkrijgen we een branching door uit Q^p kant e^p_{min} weg te laten:
$B^p = B^{p+1} \cup Q^p \setminus \{e^p_{min}\}$. Anders, stel $e^{p+1}_k \in B^{p+1}$ is gericht naar $\hat{v}^{p+1}$ in B^{p+1}, en e^{p+1}_k is ontstaan uit $e^p_k = (t,h)$, h incident met Q^p.
In Q^p is een kant, zeg e^p_r gericht naar h. Door $B^p = B^{p+1} \cup Q^p \setminus \{e^p_r\}$ te stellen, wordt het circuit Q^p verbroken en is naar h slechts één kant van B^p gericht: B^p is een branching.

N3: Als p > 0, ga naar N2.
Anders stop: B^0 is de gezochte optimale branching.

Een ALGOL-procedure van de algoritme is opgenomen in de Appendix.
Figuur 3 geeft een voorbeeld.



Figuur 3. Voorbeeld van de constructie van een optimale branching.

3.5. Efficiency en optimaliteit.

De algoritme levert een branching B^0 op, en wel op een efficiënte manier. Immers de stappen M1-M4 worden hoogstens $|V|$ keer doorlopen. Het werk in deze stappen is van de orde $|E|$. Tussen twee inkrimpingen wordt iedere kant ten hoogste één keer bekeken. De tijdsduur van de stappen N1-N3 is in de orde $|V|$. De algoritme is dus van de orde $|V| \times |E|$.
Om te bewijzen:

Stelling 4. Er is een efficiënte algoritme om een optimale branching te bepalen.

behoeven we nog slechts te laten zien dat B^0 inderdaad optimaal is. Edmonds [7] construeert een rij van duale oplossingen (u^p, y^p), $p = i, i-1, \ldots, 0$. Vervolgens toont hij inductief aan dat elk paar van oplossingen B^p en (u^p, y^p) voldoet aan de optimaliteitsvoorwaarden (26) - (28), toegepast op de graaf G^p voor $p = i, i-1, \ldots, 0$.

De algoritme gebruikt het onhanteerbare deel van de l.p.-formulering, n.l. de $2^{|V|} - |V| - 1$ voorwaarden (21), niet expliciet. Zij begint een optimale oplossing op te bouwen van l.p.-probleem (19) - (20) - (22). Zodra een voorwaarde (21) geschonden wordt, dwz. er is een circuit gevormd, hetgeen eenvoudig is vast te stellen, worden een nieuwe graaf en branching gevormd, waarvoor de voorwaarden (21) weer allen gelden.
Dat deze werkwijze tot een optimaal resultaat leidt, wordt achteraf aangetoond door de dualiteitstheorie van de lineaire programmering toe te passen op de volledige l.p.-formulering. De l.p.-formulering van dit combinatorische probleem, hoe omvangrijk ook, is zo toch van nut bij de ontwikkeling van de algoritme.

Literatuur

1. M.L. Balinski, "Labelling to obtain a maximum matching", pp. 585 - 602 in R.C. Bose and T.A. Dowling (eds.), *Combinatorial Mathematics and Its Applications*, University of North Carolina Press, Chapel Hill (1969).

2. M.L. Balinski, "Establishing the Matching Polytope", *J. Comb. Th. (B)*, 13 (1972), 1 - 13.

3. M. Bellmore and J.C. Malone, "Pathology of Traveling Salesman Subtour - Elimination Algorithms", *J. of Opns. Res.*, 19 (1971), 278 - 307.

4. M.A.H. Dempster, "Two algorithms for the Time-table Problem", pp. 63 - 85 in D.J.A. Welsh (ed.), *Combinatorial Mathematics and Its Applications*, Academic Press, London (1971).

5. J.Edmonds, "Paths, trees and flowers", *Can. J. of Mathematics*, 17 (1965), 449 - 467.

6. J. Edmonds, "Maximum Matching and a Polyhedron With 0,1-Vertices", *J. Res. Nat. Bureau of Standards*, 69 B (1965), 125 - 130.

7. J. Edmonds, "Minimum Partition of a Matroid Into Independent Subsets", *J. Res. Nat. Bureau of Standards*, 69 B (1965), 67 - 72.

8. J. Edmonds, "Optimum Branchings", *J. Res. Nat. Bureau of Standards*, 71 B (1967), 233 - 240.

9. J. Edmonds, "Systems of Distinct Representatives and Linear Algebra", *J. Res. Nat. Bureau of Standards*, 71 B (1967), 241 - 245.

10. J. Edmonds, "Submodular functions, Matroids and Certain Polyhedra", pp. 69 - 87 in R. Guy (ed.), *Combinatorial Structures and their Applications*, Gordon and Breach, New York (1970).

11. J. Edmonds, "Matroids and the greedy algorithm", *Mathematical Programming*, 1 (1971), 127 - 136.

12. J. Edmonds and D.R. Fulkerson, "Transversals and Matroid Partitions", *J. Res. Nat. Bureau of Standards*, 69 B (1965), 147 - 153.

13. J. Edmonds and D.R. Fulkerson, "Bottleneck Extrema", *J. Comb. Th.*, 8 (1970), 299 - 306.

14. J. Edmonds and E.L. Johnson, "Matching : A Well-Solved Class of Integer Linear Programs", pp. 89 - 92 in R. Guy (ed.), *Combinatorial Structures and their Applications*, Gordon and Breach, New York (1970).

15. L.R. Ford, Jr. and D.R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton (1962).

16. M. Fujii, T. Kasami and K. Ninomiya, "Optimal Sequencing of Two Equivalent Processors", *SIAM J. Appl. Math.*, 17 (1969), 784 - 789.

17. H. Gabow, "An Efficient Implementation of Edmonds' Maximum Matching Algorithm", Technical Report NO. 31, Digital Systems Laboratory, Stanford University, Stanford (1972).

18. D. Gale, "Optimal Assignments in a Ordered Set : An Application of Matroid Theory", *J. Comb. Th.*, 4 (1968), 176 - 180.

19. R.S. Garfinkel and G.L. Nemhauser, *Integer Programming*, Wiley, New York (1972).

20. A.M. Geoffrion (ed.), *Perspectives on Optimization : A Collection of Expository Articles*, Addison-Wesley, Reading (1972).

21. M. Held and R.M. Karp, "The Traveling-Salesman Problem and Minimum Spanning Trees", *J. of Opns. Res.*, 18 (1970), 1138 - 1162.

22. J.B. Kruskal, Jr., "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem", *Proc. Amer. Math. Soc.*, 7 (1956), 48 - 50.

23. S. Kundu and E.L. Lawler, "A Matroid Generalization of a Theorem of Mendelsohn and Dulmage", *Discrete Mathematics*, 4 (1973), 159 - 163.

24. E.L. Lawler, "Optimal Matroid Intersections", pp. 233 - 234 in R. Guy (ed.), *Combinatorial Structures and their Applications*, Gordon and Breach, New York (1970).

25. L. Mirsky, *Transversal Theory*, Academic Press, New York (1971).

26. P. Rosenstiehl, "L'arbre mimimum d'un graphe", in *Theory of Graphs, International Symposium Rome 1966*, Dunod/Gordon and Breach, Paris (1967).

27. W.T. Tutte, *Introduction to the Theory of Matroids*, American Elsevier Publ. Company, New York (1971).

28. D.J.A. Welsh, "Kruskal's theorem for matroids", *Proc. Camb. Phil. Soc.*, 64 (1968), 3 - 4.

29. R.J. Wilson, *Introduction to graph theory*, Oliver and Boyd, Edinburgh (1972).

Appendix : ALGOL-procedures.

```
REAL PROCEDURE OPTIMUM FOREST KRUSKAL
(M,N,TAIL,HEAD,C,TYPE,FOREST);
VALUE M,N,TYPE; INTEGER M,N,TYPE;
INTEGER ARRAY TAIL,HEAD,FOREST; REAL ARRAY C;

COMMENT OPTIMUM FOREST KRUSKAL DETERMINES IN A NON-DIRECTED GRAPH G,
        ACCORDING TO AN ALGORITHM OF J.KRUSKAL,PROC. AM. MATH.SOC.
        7(1956),48-50  !
          IF TYPE< 0 A MAXIMUM WEIGHT FOREST,
          IF TYPE= 0 A MAXIMUM WEIGHT FOREST AMONG THE FORESTS
                     WITH MAXIMUM CARDINALITY
                     (THUS IF POSSIBLE A MAXIMUM SPANNING TREE),
          IF TYPE> 0 A MINIMUM WEIGHT FOREST AMONG THE FORESTS
                     WITH MAXIMUM CARDINALITY
                     (THUS IF POSSIBLE A MINIMUM SPANNING TREE),
        INPUT ! = M, RESP N IS THE NUMBER OF VERTICES, RESP EDGES OF G,
                = EDGE J IS INCIDENT TO TAIL[J] AND HEAD[J],
                = C[J] IS THE WEIGHT OF EDGE J,
        OUTPUT! = OPTIMUM FOREST KRUSKAL, RESP FOREST[0] IS EQUAL TO
                  THE WEIGHT, RESP THE CARDINALITY OF THE SOLUTION,
                = FOREST[I], I=1,   ,FOREST[0], CONTAINS THE ADRESS OF
                  THE VERTICES, INCIDENT TO THE ITH EDGE OF THE SOLUTION,
                  INTHE ARRAYS TAIL AND HEAD;

BEGIN   INTEGER K,V,CT,CH,G,PK,CARD,DIR,M1,LAST; REAL CK;
        INTEGER ARRAY P[1:N],COMP,NEXT[1:M];

INIT!   FOR V:= 1 STEP 1 UNTIL M DO COMP[V]:= NEXT[V]:= V;
        FOR K:= 1 STEP 1 UNTIL N DO P[K]:= K;
        VEC IND QSORT(C,P,1,N);
        IF TYPE>0 THEN
        BEGIN DIR:= 1; K:= 0; LAST:= N+1 END ELSE
        BEGIN   DIR:= -1; K:= N+1; LAST:= 0; IF TYPE<0 THEN
                FOR G:= LAST+1 WHILE C[P[G]]≤0 DO LAST:= G
        END;
        CARD:= 0; CK:= 0; M1:= M-1;

MAIN PROGRAM;
        FOR K:= K+DIR WHILE K≠LAST DO
        BEGIN   PK:= P[K]; CH:= COMP[HEAD[PK]]; CT:= COMP[TAIL[PK]];
                IF CH ≠ CT THEN
                BEGIN   CARD:= CARD+1; CK:= CK+C[PK]; FOREST[CARD]:= PK;
                        IF CARD=M1 THEN GOTO OUT;
                        FOR G:= CH,NEXT[G] WHILE G≠CH DO COMP[G]:= CT;
                        G:= NEXT[CH]; NEXT[CH]:= NEXT[CT]; NEXT[CT]:= G
                END
        END;

OUT!    FOREST[0]:= CARD; OPTIMUM FOREST KRUSKAL := CK

END OPTIMUM FOREST KRUSKAL;
```

```
REAL PROCEDURE GREEDY ALGORITHM FOR
OPTIMUM TRANSVERSAL(N,M,C,L,LIST,R);
VALUE N,M; INTEGER N,M; ARRAY C; INTEGER ARRAY L,LIST,R;

COMMENT GREEDY ALGORITHM FOR OPTIMUM TRANSVERSAL COMPUTES
        A PARTIAL TRANSVERSAL OF MAXIMUM WEIGHT AMONG THE
        PARTIAL TRANSVERSALS OF MAXIMUM CARDINALITY OF
        A FAMILY OF SETS S=(S1,   ,SM), EACH SET SI
        CONSISTING OF ELEMENTS OF A SET E,
        ACCORDING TO THE GREEDY ALGORITHM,
INPUT   N IS THE NUMBER OF ELEMENTS OF SET E,
        M IS THE NUMBER OF SETS OF FAMILY S,
        C[J] IS THE WEIGHT OF ELEMENT J OF SET E,
        L[I] CONTAINS THE ADDRESS OF THE FIRST ELEMENT
        OF SET SI IN LIST, THE NEXT ADDRESSES OF LIST
        UNTIL A ZERO IS MET CONTAIN THE OTHER ELEMENTS OF SI,
OUTPUT; ELEMENT J REPRESENTS IN THE OPTIMAL SOLUTION SET R[J],
        OR DOES NOT BELONG TO THE OPTIMAL SOLUTION IF R[J]=0,
        R[0] IS EQUAL TO THE CARDINALITY OF THE SOLUTION,
        GREEDY ALGORITHM FOR OPTIMAL TRANSVERSAL IS EQUAL TO
        THE WEIGHT OF THE OPTIMAL SOLUTION;

BEGIN   INTEGER I,J,I0,J0,CARD,PJ,I1; REAL Z;
        INTEGER ARRAY LAB[1:M],P,LABJ[1:N];

        PROCEDURE LABEL(J); VALUE J; INTEGER J;
        BEGIN   INTEGER S; S:= L[J];
                FOR I0:= LIST[S] WHILE I0>0 DO
                BEGIN   IF LAB[I0]=0 THEN
                        BEGIN   LAB[I0]:= J; J0:= R[I0]
                                IF J0≠0 THEN LABEL(J0) ELSE BREAK
                        END;
                        S:= S+1
                END
        END LABEL;

        PROCEDURE BREAK;
        BEGIN   FOR I0:= I0,I1 WHILE J0≠PJ DO
                BEGIN   R[I0]:= J0:= LAB[I0];
                        I1:= LABJ[J0]; LABJ[J0]:= I0
                END;
                Z:= Z + C[PJ]; CARD:= CARD + 1;
                GOTO IF CARD<M THEN NEXTJ ELSE OUT
        END BREAK;

        FOR J:= 1 STEP 1 UNTIL N DO P[J]:= J;
        VECIND QSORT(C,P,1,N);
        CARD:= 0; Z:= 0;
        FOR I:= 1 STEP 1 UNTIL M DO R[I]:= 0;
        FOR J:= N STEP -1 UNTIL 1 DO
        BEGIN   FOR I:= 1 STEP 1 UNTIL M DO LAB[I]:= 0;
                PJ:= P[J]; LABEL(PJ);
        NEXTJ:
        END;
OUT:    R[0]:= CARD;
        GREEDY ALGORITHM FOR OPTIMUM TRANSVERSAL:= Z
END OPTIMUM TRANSVERSAL;
```

```
INTEGER PROCEDURE MAXIMUM MATCHING EDMONDS (M,N,EDGE,MATCH);
VALUE M,N; INTEGER M,N; INTEGER ARRAY EDGE,MATCH;

COMMENT MAXIMUM MATCHING EDMONDS COMPUTES A MATCHING OF
        MAXIMUM CARDINALITY OF A GRAPH G ACCORDING TO
        AN ALGORITHM OF JACK EDMONDS, CANAD.J.MATH, 17(1965),449-467,
INPUT;  M, RESP. N IS THE NUMBER OF VERTICES, RESP. EDGES OF G.
        THE VERTICES INCIDENT ON EDGE J ARE GIVEN
        IN EDGE[-J] AND EDGE[J],
OUTPUT; MAXIMUM MATCHING EDMONDS IS EQUAL TO THE NUMBER OF MATCHES,
        MATCH[I] CONTAINS THE MATCH OF VERTEX I,
        OR ZERO IF I IS EXPOSED;

BEGIN   INTEGER E,V,ORIGIN,W,K,K1,MT,LAB,EXPOSED,M1;
        BOOLEAN BACK; BOOLEAN ARRAY ACT[0:N];
        INTEGER ARRAY NEXT[-N:N],FIRST,CYCLE,MATE,LABEL[0:M+M],FLOWER[1:M+M];

        INTEGER PROCEDURE EXPAND(V); VALUE V; INTEGER V;
        BEGIN   INTEGER K,K1,V0,V1,V2,B1,B2; V0:= FLOWER[V];
                E:= FIRST[V]; MT:= ABS(MATE[V]); LAB:= ABS(LABEL[V]); B2:= 0;
                FOR V1:= V0,V2 WHILE V1 ≠ V0 DO
                BEGIN   K1:= E; V2:= CYCLE[V1];
                        E:= IF V2≠V0 THEN FIRST[V2] ELSE 0;
                        FOR K1:= K1,NEXT[K1] WHILE K1≠E DO
                        BEGIN   K:= K1; EDGE[-K]:= V1;
                                W:= EDGE[K]; ACT[ABS(K)]:= W≠V1;
                                IF W=MT THEN B1:= V1; IF W=LAB THEN B2:= V1
                        END;
                        NEXT[K]:= 0
                END;
                IF V=M1 THEN M1:= M1-1 ELSE MATE[V]:= LABEL[V]:= FIRST[V]:= 0;
                IF MT=0 THEN BEGIN B1:= V0; GOTO EX1 END;
                MATE[MT]:= LABEL[MT]:= B1; MATE[B1]:= MT; LABEL[B2]:= -LAB;
        EX1:    FOR V1:= CYCLE[B1],CYCLE[V0] WHILE V1≠B1 DO
                BEGIN   MATE[V1]:= V0:= CYCLE[V1]; MATE[V0]:= V1;
                        IF V1=B2 THEN E:= V0 ELSE IF V0=B2 THEN E:= -V1
                END;
                IF BACK ∨ E=0 THEN EXPAND:= 0 ELSE
                BEGIN   LABEL[B1]:= IF E>0 THEN -V0 ELSE -CYCLE[B1];
                        EXPAND:= E:= ABS(E); LABEL[E]:= B2
                END
        END;

        PROCEDURE SEARCH(V); VALUE V; INTEGER V;
        BEGIN   INTEGER V1;
```

```
            PROCEDURE EVEN(V,W); VALUE V,W; INTEGER V,W;
            BEGIN   INTEGER K; LABEL[V]:= W;
                    FOR K:= FIRST[V],NEXT[K] WHILE K#0 ∧ MATE[V]#0 DO
                    IF ACT[ABS(K)] THEN
                    BEGIN   W:= EDGE[K]; MT:= MATE[W]; LAB:= LABEL[W];
                            IF LAB = 0 THEN
                            BEGIN   IF MT=-10^6 THEN AUGMENT(W,V);
                                    IF MT>0 THEN
                                    BEGIN LABEL[W]:= -V; EVEN(MT,W) END
                            END     ELSE
                            IF LAB>0 ∧ MT#V THEN SHRINK(W,V)
                    END
            END EVEN;

            PROCEDURE AUGMENT(V,W); VALUE V,W; INTEGER V,W;
            BEGIN
            AM:     MATE[V]:= W; MATE[W]:= V; IF W # ORIGIN THEN
                    BEGIN V:= LABEL[W]; W:= -LABEL[V]; GOTO AM END;
                    EXPOSED:= EXPOSED - 2; GOTO OUTSEARCH
            END AUGMENT;

            PROCEDURE SHRINK(W,V); VALUE W,V; INTEGER W,V;
            BEGIN   INTEGER V0,V1,K,K1;
                    M1:= M1+1; LAB:= LABEL[W]; LABEL[W]:= FLOWER[M1]:= V;
                    FIRST[M1]:= FIRST[V]; K:= 0;
                    FOR V0:= V,V1 WHILE V0 # V DO
                    BEGIN   V1:= CYCLE[V0]:= ABS(LABEL[V0]);
                            MATE[V0]:= LABEL[V0]:= 0;
                            NEXT[K]:= K1:= FIRST[V0];
                            FOR K1:= K1,NEXT[K] WHILE K1#0 DO
                            BEGIN   K:= K1; EDGE[-K]:= M1;
                                    IF EDGE[K]=M1 THEN ACT[ABS(K)]:= FALSE
                            END
                    END;
                    IF MATE[ORIGIN]=0 THEN
                    BEGIN ORIGIN:= M1; MATE[M1]:= -10^6; EVEN(M1,M1) END ELSE
                    BEGIN MATE[M1]:= LAB; MATE[LAB]:= M1; EVEN(M1,LAB) END
            END SHRINK;

            PROCEDURE HUNGARIAN;
            BEGIN   INTEGER V,L;
            AGAIN:  FOR V:= M1 STEP -1 UNTIL M+1 DO IF LABEL[V]<0 THEN
                    BEGIN   L:= EXPAND(V); IF L#0 THEN
                            BEGIN V:= LABEL[L]; EVEN(L,V); GOTO AGAIN END
                    END
            END;

            ORIGIN:= V; FOR K:= FIRST[V],NEXT[K] WHILE K#0 DO
            IF MATE[EDGE[K]]=-10^6 THEN AUGMENT(EDGE[K],V);
            FOR V1:= 1 STEP 1 UNTIL M1 DO LABEL[V1]:= 0;
            EVEN(V,V); HUNGARIAN;

    SEPARATE: MATE[ORIGIN]:= 0; EXPOSED:= EXPOSED-1;
            FOR V1:= 1 STEP 1 UNTIL M1 DO IF LABEL[V1]#0 THEN
            BEGIN   MATE[V1]:= -MATE[V1];
                    FOR K:= FIRST[V1],NEXT[K] WHILE K#0 DO ACT[ABS(K)]:= FALSE
            END;
    OUTSEARCH;
    END SEARCH;
```

```
            EXPOSED:= M1:= M;
            FOR V:= 1 STEP 1 UNTIL M DO
            BEGIN FIRST[V]:= CYCLE[V]:= 0; MATE[V]:= -10^6 END;
            FOR E:= 1 STEP 1 UNTIL N DO
            BEGIN   ACT[E]:= TRUE; V:= EDGE[-E]; W:= EDGE[E];
                    NEXT[-E]:= FIRST[W]; FIRST[W]:= -E; CYCLE[W]:= CYCLE[W]+1;
                    NEXT[ E]:= FIRST[V]; FIRST[V]:=  E; CYCLE[V]:= CYCLE[V]+1
            END;
    INIT:   BACK:= FALSE;
            FOR V:= 1 STEP 1 UNTIL M DO IF CYCLE[V]≤1 THEN
            BEGIN   E:= CYCLE[V]; CYCLE[V]:= 10^6; EXPOSED:= EXPOSED-E-1;
                    IF E=0 THEN MATE[V]:= 0 ELSE
                    BEGIN   K:= FIRST[V];
                            FOR K1:= K WHILE ¬ACT[ABS(K)] DO K:= NEXT[K];
                            BACK:= TRUE; W:= EDGE[K]; MATE[V]:= -W; MATE[W]:= -V;
                            ACT[ABS(K)]:= FALSE; CYCLE[W]:= 10^6;
                            FOR K1:= FIRST[W],NEXT[K1] WHILE K1≠0 DO
                            IF ACT[ABS(K1)] THEN
                            BEGIN   ACT[ABS(K1)]:= FALSE;
                                    W:= EDGE[K1]; CYCLE[W]:= CYCLE[W]-1
                            END
                    END
            END;
            IF BACK THEN GOTO INIT;
            FOR V:= 1 STEP 1 UNTIL M DO IF MATE[V]=-10^6 THEN
            BEGIN   K:= FIRST[V]; FOR K1:= K WHILE ¬ACT[ABS(K)] DO K:= NEXT[K];
                    FIRST[V]:= K; FOR K1:= NEXT[K] WHILE K1≠0 DO
                    IF ACT[ABS(K1)] THEN K:= K1 ELSE NEXT[K]:= NEXT[K1]
            END;

            FOR V:= 1, V+1 WHILE EXPOSED>1 DO
            IF MATE[V]=-10^6 THEN SEARCH(V);
            BACK:= TRUE;
            FOR V:= M1 STEP -1 UNTIL M+1 DO
            IF FIRST[V]≠0 THEN EXPAND(V);
            K:= 0; FOR V:= 1 STEP 1 UNTIL M DO
            BEGIN   W:= ABS(MATE[V]); MATCH[V]:= W:= IF W=10^6 THEN 0 ELSE W;
                    IF W>V THEN K:= K+1
            END;
            MAXIMUM MATCHING EDMONDS := K
    END MAXIMUM MATCHING;
```

```
REAL PROCEDURE MAXIMUM WEIGHTED MATCHING EDMONDS (M,N,EDGE,C,MATCH);
VALUE M,N; INTEGER M,N; INTEGER ARRAY EDGE,MATCH; REAL ARRAY C;

COMMENT MAXIMUM WEIGHTED MATCHING EDMONDS COMPUTES A MATCHING OF
        MAXIMUM WEIGHT OF A GRAPH G ACCORDING TO
        AN ALGORITHM OF JACK EDMONDS, JNBRSS 69(1965),125-130,
INPUT;  M,RESP, N IS THE NUMBER OF VERTICES, RESP, EDGES OF G,
        THE VERTICES INCIDENT ON EDGE J ARE GIVEN
        IN EDGE[-J] AND EDGE[J],
        C[J] IS THE WEIGHT OF EDGE J,
OUTPUT; MAXIMUM WEIGHTED MATCHING EDMONDS IS EQUAL TO
        THE MAXIMUM WEIGHT OF A MATCHING,
        MATCH[I] CONTAINS THE MATCH OF VERTEX I,
        OR ZERO IF I IS EXPOSED, FOR I>0,
        MATCH[0] CONTAINS THE NUMBER OF MATCHES;

BEGIN   INTEGER E,V,W,ORIGIN,A,K,M,MT,LAB,M1,F;
        REAL P,Q; BOOLEAN BACK; REAL ARRAY C1[1:M+M];
        INTEGER ARRAY NEXT[-N:N],FIRST,CYCLE,MATE,LABEL,MINV[0:M+M],
                      ACT[0:N],R[-1:M+M];

        PROCEDURE EVEN(V,W); VALUE V,W; INTEGER V,W;
        BEGIN   INTEGER K; LABEL[V]:= W; IF C1[V]<-.7 THEN AUGMENT(V,V);
                FOR K:= FIRST[V],NEXT[K] WHILE K#0 ^ MATE[V]#0 DO
                IF ACT[ABS(K)]=1 THEN
                BEGIN   W:= EDGE[K]; MT:= MATE[W]; LAB:= LABEL[W];
                        IF LAB = 0 THEN
                        BEGIN   IF MT=-.6 THEN AUGMENT(W,V);
                                IF MT>0 THEN BEGIN LABEL[W]:= -V; EVEN(MT,W) END
                        END     ELSE
                        IF LAB>0 ^ MT#V THEN SHRINK(W,V)
                END
        END EVEN;

        PROCEDURE AUGMENT(V,W); VALUE V,W; INTEGER V,W;
        BEGIN   IF V=W THEN BEGIN MATE[V]:= -.6; GOTO AM1 END;
        AM:     MATE[V]:= W; MATE[W]:= V;
        AM1:    IF W#ORIGIN THEN
                BEGIN V:= LABEL[W]; W:= -LABEL[V]; GOTO AM END;
                GOTO NEWNODE
        END AUGMENT;

        PROCEDURE SHRINK(W,V); VALUE W,V; INTEGER W,V;
        BEGIN   INTEGER V0,V1,K,K1,K2,A; K:= CYCLE[ORIGIN]:= 0;
                FOR V0:= V,ABS(LABEL[V0]) WHILE V0#ORIGIN DO CYCLE[K]:= K:= V0;
                IF K#ORIGIN THEN CYCLE[K]:= ORIGIN; K:= V;
                FOR V0:= W,ABS(LABEL[V0]) WHILE TRUE DO
                BEGIN   FOR V:= CYCLE[0],CYCLE[V] WHILE V#0 DO
                        IF V=V0 THEN GOTO ROUND;
                        CYCLE[V0]:= K; K:= V0
                END;
        ROUND:  CYCLE[V0]:= K; M1:= M1+1; LAB:= LABEL[V]; Q:= -.600;
                FOR V0:= V,CYCLE[V0] WHILE V0 # V DO
                BEGIN   MATE[V0]:= LABEL[V0]:= 0;
                        IF C1[V0]<Q THEN BEGIN Q:= C1[V0]; W:= V0 END;
                        FOR K:= FIRST[V0],NEXT[K] WHILE K#0 DO EDGE[-K]:= M1
                END;
```

```
        MINV[M1]:= V0:= W; C1[M1]:= Q; P:= 0; FIRST[M1]:= K1:= FIRST[W]
SH:     FOR K1:= K1,NEXT[K1] WHILE K1≠0 DO
        BEGIN   K:= K1; K2:= ABS(K); V1:= EDGE[K];
                IF V1=M1 THEN ACT[K2]:= 2*ACT[K2] ELSE
                BEGIN   C[K2]:= C[K2]-P; IF ACT[K2]=1 THEN
                        BEGIN IF -LABEL[V1]=V0 THEN LABEL[V1]:= -M1 END
                END
        END;
        V0:= CYCLE[V0]; IF V0≠W THEN
        BEGIN NEXT[K]:= K1:= FIRST[V0]; P:= C1[V0]-Q; GOTO SH END;
        IF LAB=V THEN BEGIN ORIGIN:= M1; MATE[M1]:= -10^6; EVEN(M1,M1) END
        ELSE BEGIN MATE[LAB]:= M1; MATE[M1]:= LAB; EVEN(M1,LAB) END
END SHRINK;

PROCEDURE EXPAND(V); VALUE V; INTEGER V;
BEGIN   INTEGER K,K1,K2,V0,V1,V2,A,B1,B2; REAL CK2,CMT,CLAB;
        K1:= FIRST[V]; MT:= ABS(MATE[V]); LAB:= ABS(LABEL[V]);
        V0:= MINV[V]; Q:= C1[V0]; B2:= 0; CLAB:= CMT:= 0; E:= FIRST[V0]
        FOR V1:= V0,V2 WHILE V1≠V0 DO
        BEGIN   P:= C1[V1]-Q; K1:= E; V2:= CYCLE[V1];
                E:= IF V2≠V0 THEN FIRST[V2] ELSE 0;
                FOR K1:= K1,NEXT[K1] WHILE K1≠E DO
                BEGIN   K:= K1; EDGE[-K]:= V1; K2:= ABS(K); A:= ACT[K2];
                        IF ABS(A)≠1 THEN ACT[K2]:= A/2 ELSE
                        BEGIN   CK2:= C[K2]; C[K2]:= CK2+P; W:= EDGE[K]
                                IF W=MT  THEN BEGIN IF CK2>CMT  THEN
                                BEGIN CMT := CK2; B1:= V1 END END;
                                IF W=LAB THEN BEGIN IF CK2>CLAB THEN
                                BEGIN CLAB:= CK2; B2:= V1 END END
                        END
                END;
                NEXT[K]:= 0
        END;
        IF V=M1 THEN M1:= M1-1 ELSE MATE[V]:= LABEL[V]:= FIRST[V]:= 0;
        IF MT=0 ∨ MT=-10^6 THEN BEGIN B1:= V0; GOTO EX1 END;
        MATE[MT]:= LABEL[MT]:= B1; MATE[B1]:= MT; LABEL[B2]:= -LAB;
EX1:    FOR V1:= CYCLE[B1],CYCLE[V0] WHILE V1≠B1 DO
        BEGIN   MATE[V1]:= V0:= CYCLE[V1]; MATE[V0]:= V1;
                IF V1=B2 THEN E:= V0 ELSE IF V0=B2 THEN E:= -V1
        END;
        IF -BACK ∧ E ≠ 0 THEN
        BEGIN   LABEL[B1]:= IF E>0 THEN -V0 ELSE -CYCLE[B1];
                EVEN(ABS(E),B2)
        END
END EXPAND;

PROCEDURE MIN(C); VALUE C; REAL C; IF C≤Q THEN
BEGIN   H:= IF C<Q THEN 1 ELSE H+1; Q:= C;
        MINV[H]:= IF LAB=0 THEN K ELSE IF LAB>0 THEN V+10^4 ELSE -V-10^4
END;

M1:= M; Q:= 0; NEXT[0]:= 0;
FOR V:= 1 STEP 1 UNTIL M DO
BEGIN FIRST[V]:= 0; MATE[V]:= -10^6 END;
FOR E:= 1 STEP 1 UNTIL N DO
BEGIN   ACT[E]:= -1; V:= EDGE[-E]; W:= EDGE[E]; IF C[E]>Q THEN Q:= C[E];
        NEXT[-E]:= FIRST[W]; FIRST[W]:= -E
        NEXT[ E]:= FIRST[V]; FIRST[V]:=  E
END;
Q:= Q/2; FOR V:= 1 STEP 1 UNTIL M DO C1[V]:= IF FIRST[V]≠0 THEN Q ELSE 0;
```

```
NEWNODE;
        FOR V:= 1 STEP 1 UNTIL M1 DO IF MATE[V]=-10**6 ^ C1[V]>10**7 THEN GOTO NN;
        BACK:= TRUE; GOTO OUT;
  NN:   FOR W:= 1 STEP 1 UNTIL M1 DO LABEL[W]:= 0;
        ORIGIN:= V; EVEN(V,V);

HUNGARIAN;
        Q:= 10**600; E:= 0; F:= -1;
        FOR V:= 1 STEP 1 UNTIL M1 DO
        BEGIN   LAB:= LABEL[V]; IF LAB>0 THEN
                BEGIN   P:= C1[V]; MIN(P); LAB:= 0; R[E]:= E:= V;
                        FOR K:= FIRST[V],NEXT[K] WHILE K#0 DO
                        IF ACT[ABS(K)]=-1 THEN
                        BEGIN   W:= EDGE[K]; A:= LABEL[W]; IF A≥0 THEN
                                BEGIN   A:= IF A=0 THEN 1 ELSE
                                                IF K>0 THEN 2 ELSE 0;
                                                IF A>0 THEN MIN((P-C1[W]-C[ABS(K)])/A)
                                END
                        END
                END     ELSE IF LAB<0 THEN
                BEGIN R[F]:= F:= V; IF V>M THEN MIN(C1[MINV[V]]-C1[V]) END
        END;
        R[E]:= R[F]:= 0; FOR V:= R[0],R[V] WHILE V#0 DO C1[V]:= C1[V]-Q;
        V:= R[-1]; IF V#0 THEN FOR V:= V,R[V] WHILE V#0 DO
        BEGIN   C1[V]:= C1[V]+Q; FOR K:= FIRST[V],NEXT[K] WHILE K#0 DO
                IF ACT[ABS(K)]=1 ^ LABEL[EDGE[K]]≤0 THEN ACT[ABS(K)]:= -1
        END;
        FOR V:= 1 STEP 1 UNTIL M DO IF ABS(MINV[V])≤N THEN
        BEGIN K:= MINV[V]; MINV[V]:= EDGE[-K]; ACT[ABS(K)]:= 1 END;
        VEC QSORT(MINV,1,M);
        FOR H:= M STEP -1 UNTIL 1 DO
        BEGIN   V:= MINV[H];
                IF V>10**4 THEN BEGIN V:= V-10**4; AUGMENT(V,V) END ELSE
                IF V>-10**4 THEN EVEN(V,LABEL[V]) ELSE EXPAND(-V-10**4)
        END;
        GOTO HUNGARIAN;

OUT;    FOR V:= M1 STEP -1 UNTIL M+1 DO IF FIRST[V]#0 THEN EXPAND(V);
        Q:= 0; F:= 0; FOR V:= 1 STEP 1 UNTIL M DO
        BEGIN   W:= ABS(MATE[V]); W:= MATCH[V]:= IF W=10**6 THEN 0 ELSE W;
                IF W>V THEN
                BEGIN   K:= FIRST[V]; FOR H:= NEXT[K] WHILE EDGE[K]#W DO K:= H;
                        Q:= Q + C[ABS(K)]; F:= F+1
                END
        END;
        MATCH[0]:= F; MAXIMUM WEIGHTED MATCHING EDMONDS := Q

END MAXIMUM WEIGHTED MATCHING;
```

```
REAL PROCEDURE OPTIMUM BRANCHING EDMONDS
(M,N,TAIL,HEAD,C,TYPE,ARCSET);
VALUE M,N,TYPE; INTEGER M,N,TYPE;
INTEGER ARRAY TAIL,HEAD,ARCSET; REAL ARRAY C;

COMMENT OPTIMUM BRANCHING EDMONDS DETERMINES IN A DIRECTED GRAPH G,
        ACCORDING TO AN ALGORITHM OF JACK EDMONDS, JNBRSS Z1B(1967),233-240 :
          IF TYPE=-1 AN OPTIMUM BRANCHING, I.E. A MAXIMUM WEIGHT BRANCHING,
          IF TYPE= 0 A MAXIMUM WEIGHT BRANCHING AMONG THE BRANCHINGS
                     WITH MAXIMUM CARDINALITY
                     (THUS IF POSSIBLE A MAXIMUM SPANNING ARBORESCENCE),
          IF TYPE> 0 A MAXIMUM WEIGHT BRANCHING AMONG THE BRANCHINGS
                     WITH MAXIMUM CARDINALITY
                     HAVING NO ARC DIRECTED TOWARDS NODE TYPE
                     (THUS IF POSSIBLE A MAXIMUM SPANNING ARBORESCENCE
                     AT ROOT TYPE),
        INPUT : = M, RESP N IS THE NUMBER OF NODES, RESP ARCS OF G,
                = ARC J IS DIRECTED FROM TAIL[J] TO HEAD[J],
                = C[J] IS THE WEIGHT OF ARC J,
        OUTPUT: = OPTIMUM BRANCHING EDMONDS IS EQUAL TO
                  THE WEIGHT OF THE RESULTING ARCSET,
                = ARCSET[0] IS EQUAL TO THE CARDINALITY OF THE RESULTING ARCSET,
                = ARCSET[I] CONTAINS THE NUMBER OF THE ARC DIRECTED
                  TOWARD NODE I IN THE FORMED ARCSET, OR ZERO ELSE;

BEGIN   INTEGER K,V,CT,CH,K1,KMAX,M1; REAL P,Q,LB,CMAX,CK;
        INTEGER ARRAY BACK[-M:M],COMP,NEXT[1:M],MIN[M+1:M+M],
        CYCLE,ARC[1:M+M]; REAL ARRAY C1[1:N];

        PROCEDURE LABEL(V); VALUE V; INTEGER V;
        BEGIN   ARC[V]:= KMAX;
                CH:= COMP[HEAD[KMAX]]; CT:= COMP[TAIL[KMAX]];
                IF CH=CT THEN SHRINK(V) ELSE
                BEGIN   FOR K:= CH,NEXT[K] WHILE K#CH DO COMP[K]:= CT;
                        K:= NEXT[CH]; NEXT[CH]:= NEXT[CT]; NEXT[CT]:= K
                END
        END LABEL;

        PROCEDURE SHRINK(W); VALUE W; INTEGER W;
        BEGIN   INTEGER V,V0,VC,VMIN;
                VC:= M1:= M1+1; Q:= .600; V0:= W;
                FOR K:= KMAX,ARC[W] WHILE K # KMAX DO
                BEGIN   P:= C1[K]; IF P<Q THEN BEGIN Q:= P; VMIN:= W END;
                        V0:= ARCSET[TAIL[K]]; IF V0≤M THEN ARCSET[V0]:= M1 ELSE
                        BEGIN   CYCLE[VC]:= V:= CYCLE[V0];
                                FOR V:= V,CYCLE[V] WHILE V # 0 DO
                                BEGIN ARCSET[V]:= M1; VC:= V END
                        END;
                        CYCLE[V0]:= W; W:= V0
                END;
                MIN[M1]:= VMIN; K:= -M1; CMAX:= LB;
```

```
                    FOR W:= VMIN,CYCLE[W] WHILE W ≠ VMIN DO
                    BEGIN   IF W≤M THEN CYCLE[VC]:= VC:= W; P:= C1[ARC[W]]-Q;
                            FOR K1:= BACK[-W],BACK[K1] WHILE K1 ≠ 0 DO
                            IF ARCSET[TAIL[K1]]≠M1 THEN
                            BEGIN   BACK[K]:= K:= K1; C1[K]:= CK:= C1[K]-P;
                                    IF CK>CMAX THEN BEGIN CMAX:= CK; KMAX:= K END
                            END
                    END;
                    CYCLE[VC]:= BACK[K]:= 0;
                    IF CMAX>LB THEN LABEL(M1) ELSE ARC[M1]:= 0
            END SHRINK;

            PROCEDURE EXPAND(V); VALUE V; INTEGER V;
            BEGIN   K:= ARC[V];
                    ARC[IF K=0 THEN MIN[V] ELSE HEAD[K]]:= K
            END EXPAND;

    INIT:   FOR V:= 1 STEP 1 UNTIL M DO
            BEGIN   BACK[-V]:= 0;
                    ARCSET[V]:= COMP[V]:= NEXT[V]:= V
            END;
            FOR K:= 1 STEP 1 UNTIL N DO
            BEGIN   C1[K]:= C[K]; V:= HEAD[K];
                    BACK[K]:= BACK[-V]; BACK[-V]:= K
            END;
            IF TYPE<0 THEN BEGIN LB:= -.500; TYPE:= 0; END ELSE
            BEGIN LB:= -.500; IF TYPE>0 THEN ARC[TYPE]:= 0 END;
            M1:= M;

    BRANCHING;
            FOR V:= TYPE-1 STEP -1 UNTIL 1, TYPE+1 STEP 1 UNTIL M DO
            BEGIN   K:= -V; CMAX:= LB;
                    FOR K:= BACK[K] WHILE K ≠ 0 DO
                    BEGIN   CK:= C1[K]; IF CK>CMAX THEN
                            BEGIN CMAX:= CK; KMAX:= K END
                    END;
                    IF CMAX>LB THEN LABEL(V) ELSE ARC[V]:= 0
            END;
            V:= M+1;
            FOR M1:= M1 STEP -1 UNTIL V DO EXPAND(M1);

    OUT:    CMAX:= 0; KMAX:= 0;
            FOR V:= 1 STEP 1 UNTIL M DO
            BEGIN   ARCSET[V]:= K:= ARC[V]; IF K>0 THEN
                    BEGIN CMAX:= CMAX+C[K]; KMAX:= KMAX+1 END
            END;
            ARCSET[0]:= KMAX; OPTIMUM BRANCHING EDMONDS := CMAX

END OPTIMUM BRANCHING;
```