stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNDE BN 21/73 AUGUSTUS

B. DORHOUT
HET LINEAIRE TOEWIJZINGSPROBLEEM:
VERGELIJKING VAN ALGORITMEN

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): 90B10, 90C05

Inhoud

		blz.
1.	Inleiding	1
2.	Algoritmen	2
3.	Vergelijking van de prestaties van enige algoritmen	13
	Literatuur	16
	Appendix: ALGOL-procedure voor de verbeterde algoritme van Tomizawa	18

1. Inleiding

Onder het *(lineaire) toewijzingsprobleem* verstaat men het volgende probleem: Minimaliseer

$$(1.1) \qquad \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_{ij}$$

onder de bijvoorwaarden

$$(1.2) \qquad \sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1,\ldots,n,$$

$$(1.3) \qquad \sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1,\ldots,n,$$

$$(1.4) \qquad x_{ij} \geq 0, \qquad i = 1,\ldots,n;\ j = 1,\ldots,n.$$

Omdat er onder de optimale oplossingen van dit probleem altijd tenminste één is, die voldoet aan

$$(1.5) \qquad x_{ij} = 0 \text{ of } 1 \qquad i = 1,\ldots,n;\ j = 1,\ldots,n$$

(zie bijvoorbeeld [2]), zullen wij ook problemen, waarin (1.4) door (1.5) is vervangen, lineaire toewijzingsproblemen noemen.

Voor een oplossing, die aan (1.5) voldoet, geldt steeds, dat bij elke i juist één j behoort, waarvoor $x_{ij} = 1$ is en bij elke j juist één zo'n i. Met een dergelijke oplossing correspondeert dus steeds een permutatie van de getallen 1,...,n. Het beschreven model kan dan ook gebruikt worden voor de formulering van verschillende volgordeproblemen, zoals dienstregelingsproblemen. Zie hiervoor bijvoorbeeld [18].

Een ander voorbeeld is het personeelstoewijzingsprobleem, o.a. beschreven

door Votaw en Orden ([19],[20]). Bij een dergelijk probleem wordt gevraagd n functies aan n personen toe te wijzen, waarbij elke persoon juist één functie moet vervullen. Als de kosten, die voortvloeien uit het toewijzen van functie i aan persoon j, alleen afhangen van de combinatie van i en j, kunnen zij worden voorgesteld door een getal a_{ij}. Voert men dan variabelen x_{ij} in, die weergeven, of persoon j in functie i wordt aangesteld ($x_{ij}=1$) of niet ($x_{ij}=0$), dan zorgen de bijvoorwaarden (1.2), (1.3) en (1.5) er voor, dat aan de gestelde eisen wordt voldaan, terwijl (1.1) de totale kosten weergeeft, die men wil minimaliseren.

Naast de directe toepassingen van het toewijzingsprobleem speelt het een rol als relaxatie van moeilijker oplosbare problemen ([8]).

Daar de in de praktijk voorkomende lineaire toewijzingsproblemen dikwijls zeer groot zijn, zoals bij de toepassing op dienstregelingsproblemen, of een grote reeks van toewijzingsproblemen moet worden opgelost, zoals bij de toepassing als relaxatie van een moeilijker probleem, is het van groot belang over een efficiënte algoritme voor het oplossen van (1.1),...,(1.4) te beschikken. Daar komt nog bij, dat het toewijzingsprobleem kan worden gezien als specialisatie van het transportprobleem, en omgekeerd het transportprobleem als specialisatie van het toewijzingsprobleem kan worden opgevat (vgl. Dantzig [2]). Conclusies aangaande algoritmen voor het toewijzingsprobleem zullen dus in het algemeen tevens van belang zijn voor corresponderende algoritmen voor het transportprobleem en zijn generalisaties, zoals de verschillende kostenstromingsproblemen (zie [7]).

2. Algoritmen

De meeste algoritmen voor het oplossen van (1.1),...,(1.4) maken gebruik van de zogenaamde "complementary slackness" eigenschap voor lineaire programmeringsproblemen. Bij de formulering van deze eigenschap nemen we eerst aan, dat (1.3) met -1 is vermenigvuldigd. Het bij (1.1),...,(1.4) behorende duale probleem luidt dan:

maximaliseer

$$(2.1) \qquad -\sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j$$

onder de bijvoorwaarden

$$(2.2) \qquad -u_i + v_j \leq a_{ij}, \qquad i = 1,\ldots,n;\ j = 1,\ldots,n.$$

De "complementary slackness" eigenschap luidt nu als volgt:

Stelling 2.1: Een oplossing van (1.1),...,(1.4) is dan en slechts dan optimaal, als u_i- en v_j-waarden gevonden kunnen worden, die aan (2.2) voldoen en waarvoor geldt:

$$(2.3) \qquad u_i + a_{ij} - v_j > 0 \Longrightarrow x_{ij} = 0, \qquad i = 1,\ldots,n;\ j = 1,\ldots,n.$$

De algoritmen voor (1.1),...,(1.4) kunnen bijna alle worden ingedeeld in drie klassen: men onderscheidt *primale*, *duale* en *gemengde* "primal-dual" algoritmen. *) Al deze algoritmen zijn iteratief en eindigen, zodra een oplossingenpaar voor het oorspronkelijke en het duale probleem gevonden is, dat aan (1.2), (1.3), (1.4), (2.2) en (2.3) voldoet (vgl. stelling 2.1). De drie benaderingswijzen verschillen in de verzameling van bijvoorwaarden, waaraan in de niet-optimale stappen niet voldaan wordt, en in het criterium, waarmee de kwaliteit van een niet-optimaal oplossingenpaar wordt gemeten. Bij de primale algoritmen wordt niet voldaan aan (2.2) en wordt een rij van oplossingen geconstrueerd, waarvoor de waarde van (1.1) monotoon niet-stijgend is, bij de duale algoritmen wordt niet voldaan aan (1.4) en is de waarde van (2.1) monotoon niet-dalend. Bij de gemengde algoritmen worden (1.2) en (1.3) vervangen door

$$(2.4) \qquad \sum_{j=1}^{n} x_{ij} \leq 1, \qquad i = 1,\ldots,n$$

---

*) De branch-and-bound methode, beschreven in het boek van Hillier en Lieberman [9], valt buiten deze indeling. Deze methode is waarschijnlijk slechts om didactische redenen in het boek opgenomen.

resp.

$$(2.5) \qquad \sum_{i=1}^{n} x_{ij} \leq 1, \qquad j = 1,\ldots,n,$$

en zijn de waarden van

$$(2.6) \qquad w = \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij}$$

monotoon niet-dalend.

Bij de vergelijking van de verschillende algorithmen hebben wij getracht dubbel werk te vermijden door ons te beperken tot het onderzoek van die algoritmen, waarvan nog niet door anderen was vastgesteld, dat zij onbevredigend werkten. Volledigheidshalve vermelden wij eerst enige conclusies, die uit de bestudering der literatuur getrokken kunnen worden.

a. De zogenaamde Hongaarse methode, een gemengde algoritme, afkomstig van Kuhn [11], werd lange tijd als de beste methode beschouwd. Vooral de modificatie van Munkres [12], zoals beschreven [13] en geprogrammeerd [14] door Silver, werd bekend. Later ontstonden versies, die meer aansloten bij de theorie van de stromingen in netwerken volgens Ford en Fulkerson [7]. Naast de procedure van Silver [14], die wij hebben overgenomen en beproefd, werd een verbeterde versie geprogrammeerd.

b. Er zijn slechts drie primale methoden tot de literatuur doorgedrongen: de simplexmethode, afkomstig van Dantzig (zie bijvoorbeeld [2]), een methode van Balinski en Gomory [1] en de negatieve cykel methode van Klein [10]. Daar de laatstgenoemde methoden in een vergelijkende studie van Florian en Klein [6] veel slechtere resultaten opleverden dan de Hongaarse methode, hebben wij deze methoden niet verder onderzocht, wel de simplexmethode.

c. Duale algoritmen zijn niet of nauwelijks bekend.

d. In de laatste tijd zijn twee gemengde algoritmen gepubliceerd door Tomizawa [17] resp. Tabourier [16], die nog niet met andere algoritmen waren vergeleken. Deze algoritmen zijn door ons geprogrammeerd en getest.

Een synthese van de beide algoritmen bleek een aanzienlijk beter resultaat op te leveren dan de algoritmen in hun oorspronkelijke versie.

Beschrijving van de onder d genoemde algoritmen

Wij gaan uit van een gericht netwerk met punten $P_0,\ldots,P_n,Q_0,\ldots,Q_n$. De pijlen van het netwerk lopen van P_0 naar $P_1,\ldots,P_n$, van $P_1,\ldots,P_n$ naar alle $Q_1,\ldots,Q_n$ en van $Q_1,\ldots,Q_n$ naar Q_0. De pijlen (P_0,P_i), $i = 1,\ldots,n$, en (Q_j,Q_0), $j = 1,\ldots,n$, hebben een capaciteit 1 en kosten 0, de pijlen (P_i,Q_j) hebben capaciteit ∞ en kosten a_{ij}, $i = 1,\ldots,n$; $j = 1,\ldots,n$.

De sterkte van een stroom door (P_i,Q_j) wordt weergegeven door x_{ij}. Een toegelaten stroom door het netwerk correspondeert met een oplossing van (2.4), (2.5), (1.4). Wij zullen slechts stromen beschouwen, die voldoen aan (1.5). P_i zullen wij verzadigd noemen, als $x_{ij} = 1$ voor zekere j, en anders onverzadigd.

Verder krijgen wij te maken met wegen van de gedaante $(P_0,P_{i_1}),(P_{i_1},Q_{j_1}),(Q_{j_1},P_{i_2}),\ldots,(P_{i_r},Q_{j_r}),(Q_{j_r},Q_0)$, waarin de notatie (Q_j,P_i) weergeeft, dat langs deze weg de pijl (P_i,Q_j) tegen de pijlrichting in doorlopen wordt. Een weg heeft verder de eigenschap, dat geen enkel punt er vaker dan een maal in voorkomt. Een *stroomvermeerderende weg*, behorend bij een oplossing van (2.4), (2.5), (1.4) is een weg van de zojuist aangegeven gedaante, waarvoor geldt:

$$\sum_{j=1}^{n} x_{i_1 j} = 0, \tag{2.7}$$

$$x_{i_k j_k} = 0, \qquad k = 1,\ldots,r, \tag{2.8}$$

$$x_{i_{k+1} j_k} = 1, \qquad k = 1,\ldots,r-1,$$

$$\sum_{i=1}^{n} x_{i j_r} = 0. \tag{2.9}$$

Een nieuwe oplossing van (2.4), (2.5), (1.4) verkrijgt men, als de door (2.8) vastgelegde waarden worden veranderd in

$$x_{i_k j_k} = 1, \qquad k = 1,\ldots,r,$$

$$x_{i_{k+1} j_k} = 0, \qquad k = 1,\ldots,r. \tag{2.10}$$

Door deze omzetting neemt (2.6) met 1 eenheid toe en (1.1) met

$$\sum_{k=1}^{r} a_{i_k j_k} - \sum_{k=1}^{r-1} a_{i_{k+1} j_k}. \tag{2.11}$$

Noemt men a_{ij} de lengte van (P_i,Q_j) en $-a_{ij}$ de lengte van (Q_j,P_i), dan stelt (2.11) de lengte van de beschouwde stroomvermeerderende weg voor. De bij een oplossing van (2.4), (2.5), (1.4) behorende stroomvermeerderende weg, waarvoor (2.11) minimaal is, zullen wij een minimale stroomvermeerderende weg noemen.

Wij maken nu gebruik van een stelling van Jewell opgenomen als stelling 3.5 in het boek van Ford en Fulkerson ([7], blz. 121). Voor de beschreven situatie luidt deze:

Stelling 2.2: Levert een oplossing van (2.4), (2.5), (2.6), (1.4) de minimale waarde van (1.1) op onder de bijwaarde $w = \bar{w}$, dan ontstaat door de overgang van (2.8) naar (2.10) een optimale oplossing onder de bijvoorwaarde $w = \bar{w} + 1$.

Uit deze stelling volgt, dat (1,1),...,(1.4) kan worden opgelost door middel van de volgende

basisalgoritme:

1. $x_{ij} := 0, \quad i = 1,\ldots,n;\ j = 1,\ldots,n.$ (w=0).
2. Bepaal een minimale stroomvermeerderende weg van P_0 naar Q_0 en ga over van (2.8) op (2.10) (w:=w+1). Herhaal deze stap tot w = n is.

De volgende van de Hongaarse methode bekende stelling is verder van belang.

Stelling 2.3: Worden alle elementen van een rij i (kolom j) van de afstandsmatrix $A = (a_{ij})$ met eenzelfde bedrag b vermeerderd, dan heeft het hierdoor gewijzigde toewijzingsprobleem dezelfde oplossing als het oorspronkelijke. De optimale waarde van (1.1) neemt door de wijziging toe met b.

Uit stelling 2.3 volgt, dat men kan beginnen met het optellen van een groot getal M bij alle rijen van A behalve de eerste. Bij de eerste stap van de basisalgoritme wordt dan een minimale stroomvermeerderende weg gevonden, die begint met (P_0,P_1). Vervolgens kan men M aftrekken van de tweede rij van A, zodat bij de tweede toepassing van stap 2 de minimale stroomvermeerderende weg begint met (P_0,P_2). Dit gedachtenexperiment voortzettend komt men tot de conclusie, dat men in stap 2 de eerste pijl van een stroomvermeerderende weg willekeurig kan kiezen. Deze eigenschap wordt zowel door Tabourier als door Tomizawa toegepast. Tomizawa kiest bovendien de laatste pijl van de stroomvermeerderende weg nog willekeurig, zodat hij bij de s-de toepassing van stap 2 slechts te maken heeft met een s×s-deelmatrix van A.

Een ander gevolg van stelling 2.3 is, dat men A mag *reduceren*. In een rijreductie wordt elke rij van A met minimum verminderd, waarna op analoge wijze een kolomreductie wordt uitgevoerd. Na deze bewerkingen zijn alle elementen van A niet-negatief en bevatten elke rij en elke kolom tenminste één nul. Men behoeft dan in de eerste uitvoeringen van stap 2 geen kortste wegen te berekenen, daar elke weg (P_0,P_i), (P_i,Q_j), (Q_j,Q_0) met $a_{ij} = 0$ de minimale lengte nul heeft. Toepassing van deze eigenschap stelt Tabourier in staat bij de eerste niet-triviale uitvoering van stap 2 van dezelfde beginoplossing uit te gaan, als waarmee de Hongaarde methode begint. Hiermee is de algoritme van Tabourier beschreven. Voor de berekening van de kortste wegen wordt de algoritme van Ford gebruikt (zie bijv. [7]).

Tomizawa transformeert A bij elke stap 2 zodanig, dat alle elementen van de te onderzoeken s×s-deelmatrix van A aan het eind van stap 2 voldoen aan

(2.12) $a_{ij} \geqq 0, \qquad i = 1,\ldots,s;\ j = 1,\ldots,n,$

(2.13) $a_{ij}x_{ij} = 0, \qquad i = 1,\ldots,n;\ j = 1,\ldots,n.$

Hij maakt hierbij gebruik van de algoritme van Dijkstra [3] voor het bepalen van kortste routes in netwerken met niet-negatieve afstanden.

In een netwerk met punten $P_1,\ldots,P_n$ kunnen volgens Dijkstra de kortste afstanden van P_1 naar $P_2,\ldots,P_n$ als volgt berekend worden:

1. Kies $m \in \{g \mid a_{1g} = \min a_{1j}\}$.
   Geef elk punt P_j, $j = 1,\ldots,n$, een merk $[\lambda_j,d_j]$, met

$$\lambda_j = \begin{cases} 0 & j = 1 \\ -1 & j \neq 1 \end{cases}$$

$$d_j = \begin{cases} 0 & j = 1 \\ a_{1j} & j \neq 1 \end{cases}$$

2. $\lambda_m := -\lambda_m$.
   Bereken d_m+a_{mj} voor alle j met $\lambda_j < 0$.
   Is $d_m+a_{mj} < d_j$, dan wordt $\lambda_j := -m$, $d_j := d_m+a_{mj}$.
   Kies $m \in \{g \mid d_g = \min d_j\}$.
   Herhaal deze stap tot $\lambda_j \geqq 0$, $j = 1,\ldots,n$.

Toelichting: In het merk $[\lambda_j,d_j]$ geeft d_j telkens de tot dusver gevonden kortste afstand tot P_j weer. Voor $\lambda_j < 0$ kan eventueel in een latere stap nog een lagere d_j-waarde gevonden worden, voor $\lambda_j > 0$ is d_j de definitieve kortste afstand. $|\lambda_j|$ stelt de index voor van het laatste punt op de tot dusver gevonden kortste route van P_1 naar P_j.
De achtereenvolgens in stap 2 gevonden definitieve d_m-waarden ($\lambda_m := -\lambda_m$) vormen een monotoon niet-dalende getallenrij. Beëindigt men de algoritme al, zodra $\lambda_n > 0$, dan geldt

(2.14) $\lambda_j < 0 \Longrightarrow d_j \geq d_n$.

Stel nu, dat voor s = r-1 een oplossing $x_{h_1 1} = x_{h_2 2} = \ldots = x_{h_{r-1} r-1} = 1$ en een matrix A gevonden zijn, die aan (2.12) en (2.13) voldoen. Volgens de algoritme van Tomizawa verloopt de r-de uitvoering van stap 2 van de basis-algoritme dan als volgt.

a. Reduceer rij r van A, als deze rij negatieve elementen bevat. Dan is voldaan aan (2.12) voor s = r.

b. Laat de punten $P_{r+1},\ldots,P_n,Q_{r+1},\ldots,Q_n$ en alle met deze punten verbonden pijlen weg en bepaal met de algoritme van Dijkstra de kortste afstanden d_j vanuit P_0 (dus P_r) naar de punten Q_j, totdat de kortste afstand d_r tot Q_r gevonden is (wegens (2.13) is steeds d_j tevens de kortste afstand tot P_{h_j}).

c. $\Delta u_{h_j} := \Delta v_j := \begin{cases} \min(d_j,d_r) & j = 1,\ldots,r, \\ 0 & j = r+1,\ldots,n. \end{cases}$

d. $a_{ij} := a_{ij} + \Delta u_i - \Delta v_j$ $\qquad i = 1,\ldots,n;\ j = 1,\ldots,n.$

e. Ga over van (2.8) op (2.10).

Bewijs, dat aan het eind van de stap wederom (2.13) en (2.14) gelden:
Stel, dat bij voortzetting van de algoritme van Dijkstra de kortste afstanden $\bar{d}_j$ tot alle punten $Q_1,\ldots,Q_r$ gevonden waren. Dan had men kunnen stellen

$$\bar{\Delta}u_{h_j} := \bar{\Delta}v_j := \bar{d}_j, \qquad j = 1,\ldots,r$$

en gold

(2.15) $$a_{ij} + \bar{\Delta}u_i - \bar{\Delta}v_j \geq 0, \qquad i = 1,\ldots,r;\ j = 1,\ldots,r$$

want was dit niet het geval geweest, dan had er een kortere weg naar Q_j bestaan. Er geldt echter

$$\Delta u_i = \min(\bar{\Delta} u_i, \bar{d}_r), \qquad i = 1,\ldots,r,$$

$$\Delta v_j = \min(\bar{\Delta} v_j, \bar{d}_r), \qquad j = 1,\ldots,r.$$

Noemt men de verzameling van de punten, waarvoor de definitieve kortste afstand bepaald is, K, en geeft men het complement van K t.o.v. $\{P_1,\ldots,P_r,Q_1,\ldots,Q_r\}$ weer door $\bar{K}$, dan is duidelijk

$$(2.16) \qquad a_{ij} + \Delta u_i - \Delta v_j \geq 0$$

voor $i = 1,\ldots,r$; $j = r+1,\ldots,n$, wegens (2.12), $\Delta u_i \geq 0$, $\Delta v_j = 0$,
voor alle i en j met $P_i \in K$, $Q_j \in K$, wegens (2.15), $\Delta u_i = \bar{\Delta} u_i$, $\Delta v_j = \bar{\Delta} v_j$,
voor alle i en j met $P_i \in \bar{K}$, $Q_j \in K$, wegens (2.12) $\Delta u_i = \bar{d}_r$, $\Delta v_j \leq \bar{d}_r$,
voor alle i en j met $P_i \in K$, $Q_j \in \bar{K}$, wegens (2.15), $\Delta u_i = \bar{\Delta} u_i$, $\Delta v_j \leq \bar{\Delta} v_j$,
en voor alle i en j met $P_i \in \bar{K}$, $Q_j \in \bar{K}$, wegens (2.12), $\Delta u_i = \bar{d}_r$, $\Delta v_j = \bar{d}_r$.

Verder blijkt uit de procedure van Dijkstra, dat voor de pijlen (P_i,Q_j) op de kortste weg van P_r naar Q_r steeds geldt

$$\Delta v_j = a_{ij} + \Delta u_i,$$

zodat (2.13) geldt voor alle in (2.8) en (2.10) vermelde x_{ij}-waarden. Voor de oude toewijzingen $x_{ij} = 1$ blijft (2.13) gelden wegens $a_{ij} = 0$ en $\Delta u_{h_j} = \Delta v_j$.

Daar de algoritme begint met $x_{ij} = 0$, $i = 1,\ldots,n$; $j = 1,\ldots,n$, volgt uit het bovenstaande, dat de algoritme van Dijkstra op de beschreven wijze kan worden toegepast.

Een vergelijking van de algoritmen van Tabourier en Tomizawa valt sterk in het voordeel van de laatste uit. Het is immers bekend ([4],[5]), dat voor de berekening van de kortste afstand tussen twee punten in een netwerk met n punten $O(n^2)$ berekeningen nodig zijn, als de afstanden niet-negatief zijn, maar $O(n^3)$, als dit niet het geval is. Bovendien heeft men bij de algoritme van Tabourier steeds te maken met een netwerk met n punten en bij de algo-

ritme van Tomizawa bij de s-de uitvoering van stap 2 van de basisalgoritme met een netwerk met s punten (in feite spelen slechts de punten Q_j een rol).

Een eerste verbetering van de algoritme van Tomizawa kan verkregen worden door op de wijze van Tabourier uit te gaan van de beginoplossing van de Hongaarse methode.
Een tweede verbetering verkrijgt men door evenals Tabourier in stap 2 de eindpijl van de stroomvermeerderende weg vrij te laten. Weliswaar moet dan in stap 2 telkens een netwerk met n knooppunten Q_j worden onderzocht, maar men kan de algoritme van Dijkstra beëindigen, zodra van één van de onverzadigde Q_j de definitieve afstand berekend is.

Het bewijs van de toepasbaarheid van de algoritme van Dijkstra kan voor de gewijzigde algoritme vrijwel onveranderd worden overgenomen.

Voorbeeld van de oplossing van een probleem d.m.v. de gewijzigde algoritme van Tomizawa.

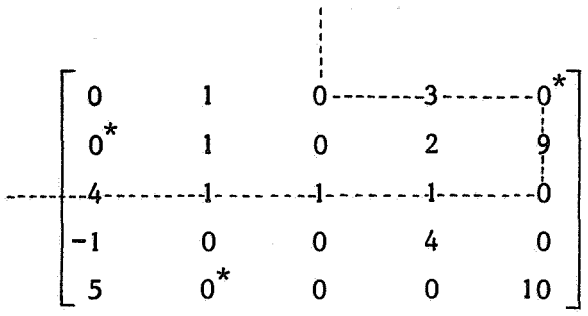
Zij $A = \begin{bmatrix} 7 & 12 & 9 & 11 & 5 \\ 5 & 10 & 7 & 8 & 12 \\ 14 & 15 & 13 & 12 & 8 \\ 8 & 13 & 11 & 14 & 7 \\ 10 & 9 & 7 & 6 & 13 \end{bmatrix}$.

Na reductie ontstaat de matrix uit tableau 2.1. De beginoplossing is met sterretjes aangegeven.

λ_j	~~-3~~ -1	-3	~~-3~~ -1	-3	~~-3~~ +3
Δu_i \ d_j	~~6~~ 2	4	~~4~~ 3	4	0
0	2	4	3	6	0^*
2	0^*	2	1	3	7
0	6	4	4	4	0
	1	3	3	7	0
	4	0^*	0	0	7

Tableau 2.1: Beginoplossing

Toelichting bij de Δu_i-kolom en de λ_j en d_j-rijen:
Begonnen wordt met het vermelden van $\Delta u_3 = 0$ (rij 3 correspondeert met het eerste niet verzadigde punt P_i). Daarna worden de voorlopige waarden van λ_j en d_j, $j = 1,\ldots,5$, vermeld: $\lambda_j := -3$, $d_j := a_{3j}$. Daar $d_5 = \min_{j=1,\ldots,5} d_j$ wordt vervolgens $\lambda_5 := 3$, $\Delta u_1 := d_5$. Nagegaan wordt, of d_j, $j = 1,\ldots,4$, verlaagd kunnen worden. Dit is het geval voor $j = 1$ en $j = 3$: $d_1 := 0+2$, $d_3 := 0+3$. Daar nu $d_1 = \min_{j=1,\ldots,4} d_j$ wordt $\lambda_1 := -\lambda_1$, $\Delta u_2 := d_1$. In de d_j-rij vinden hierna geen veranderingen meer plaats. $d_3 = \min_{j=2,3,4} d_j$, met Q_3 onverzadigd, zodat de algoritme van Dijkstra afgebroken kan worden.
Uit de d_j-rij volgt nu:
$\Delta v_1 = \Delta u_2 = 2$, $\Delta v_2 = \Delta u_5 = 3$, $\Delta v_3 = 3$, $\Delta v_4 = 3$, $\Delta v_5 = \Delta u_1 = 0$. Verder was uitgegaan van $\Delta u_3 = 0$ en heeft rij 4 nog geen enkele rol gespeeld: $\Delta u_4 = 0$.
De onder d vermelde transformatie levert

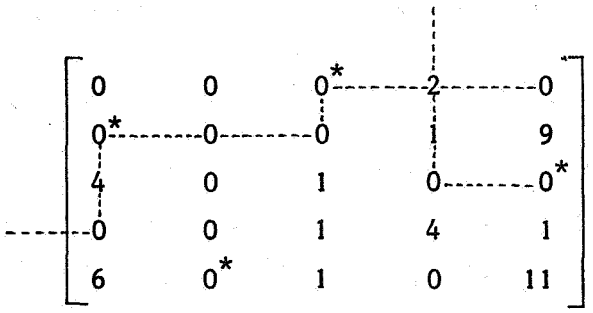


D.m.v. de λ_j in tableau 2.1 kan de aangegeven stroomvermeerderende weg worden afgelezen. Na de verandering van de oplossing begint de volgende stap met de reductie van rij 4. Het eindtableau wordt

λ_j	4	-4	2	-3	1
d_j / Δu_j	0	1	0	1	0
0	0	1	0*	3	0
0	0*	1	0	2	9
0	4	1	1	1	0*
0	0	1	1	5	1
	5	0*	0	0	10

Tableau 2.2

Na transformatie ontstaat



zodat de oplossing van het probleem wordt: $x_{41} = x_{52} = x_{23} = x_{34} = x_{15} = 1$.

Opmerking

Uit de beschrijving van de algoritme volgt, dat de som van de Δu_i en Δv_j, die in de achtereenvolgende stappen zijn berekend, de optimale waarden van de duale variabelen opleveren. Aangezien in het eindtableau aan (1.2), (1.3), (1.4), (2.2), (2.3) is voldaan, was voor het bewijs van de geldigheid van de algoritme van Tomizawa toepassing van stelling 2.2 niet nodig geweest.

3. Vergelijking van de prestaties van enige algoritmen

Voor verschillende algoritmen werden ALGOL-programma's geschreven, die op de EL-X8 installatie van het Mathematisch Centrum werden beproefd. Enige rekentijden (in sec.) zijn vermeld in tabel 3.1. De probleemnummers zijn van de gedaante abcd, waarbij ab de waarde van n aangeeft. Verder geldt bij

c = 0 : a_{ij} aselect, geheel, $0 \leq a_{ij} < 10$
c = 1 : a_{ij} aselect, geheel, $0 \leq a_{ij} < 50$
c = 2 : a_{ij} aselect, geheel, $0 \leq a_{ij} < 250$
c = 3 : a_{ij} aselect, $0 \leq a_{ij} < 1$
c = 4 : a_{ij} ontleend aan dienstregelingsprobleem, geheel, $0 \leq a_{ij} < 40$
c = 5 : gelijk aan a_{ij} bij c = 4, vermeerderd met aselect getal ε_{ij}, $0 \leq \varepsilon_{ij} < 0{,}01$.

De kolomopschriften geven weer:

H1 : Hongaarse methode volgens [14].

H2 : Verbeterde algoritme voor de Hongaarse methode, transformaties in de A-matrix worden uitgevoerd.

H3 : Eenvoudiger versie van H2, waarbij geen transformaties worden uitgevoerd, maar de duale variabelen worden bijgewerkt.

S : Simplexmethode volgens de standaardprocedure voor het transportprobleem, A-matrix opgeslagen in het trommelgeheugen, wordt daarvan rij voor rij naar het kerngeheugen overgebracht.

Ta : Tabourier

To : Tomizawa, uitgaande van de beginoplossing van de Hongaarse methode, duale variabelen worden bijgewerkt.

V : Verbeterde algoritme van Tomizawa, duale variabelen worden bijgewerkt.

Uit tabel 3.1 blijkt, dat de Hongaarse algoritmen gunstiger werken naarmate het probleem sterker duaal ontaard is, terwijl de andere algoritmen hiervoor ongevoelig zijn. Verder blijkt het ongunstig te zijn om matrices te transformeren.
Ten opzichte van de andere algoritmen vallen bij toenemende n de rekentijden voor de Hongaarse algoritmen in verhouding steeds slechter uit.

Niet in tabel 3.1 opgenomen resultaten tonen aan, dat trommelgebruik bij de Hongaarse algoritmen leidde tot een toename van de rekentijd met gemiddeld 50%, terwijl deze toename bij de simplexmethode bij toenemende n steeds verder afnam (bij n=75 gemiddeld 2,5 sec.).

Daar weinig geexperimenteerd is met de programmering van de verbeterde algoritme van Tomizawa, kan men verwachten, dat met deze algoritme zeer gunstige resultaten te bereiken zijn. Ook de simplexmethode behoeft echter nog niet geheel te worden afgeschreven, daar recent enige ideeën over de programmering van deze methode zijn ontwikkeld ([15]), die mogelijk betere resultaten zullen opleveren.

probleem \ algoritme	H1	H2	H3	S	Ta	To	V
2501		1,85	1,92	11,74	4,41	2,70	2,14
2502		1,57	1,66	10,03	4,33	2,48	2,25
2503		2,03	2,10	9,22	4,79	2,59	2,21
2511		2,96	2,53	10,85	7,53	3,47	2,20
2512		2,32	1,95	8,62		2,88	1,74
2513		2.71	2,12	12,27		2,78	1,97
2521		4,19		11,64		2,40	1,62
2522		4,71		9,72		2,51	2,11
2523		2,56		8,95		2,24	1,39
2531		4,08		11,62		3,24	1,81
2532		4,33		10,02		2,96	1,82
2533		2,72		10,08		3,43	1,66
5001		4,50	5,33	19,73		12,94	10,96
5002		5,84		25,94		12,65	10,49
5003		6,19		21,50		12,98	11,44
5011		12,03		31,74		16,80	7,88
5012		13,17		37,56		17,51	9,79
5013		10,12		34,07		17,17	8,67
5021		22,72	18,29	33,11		20,60	9,34
5022	35,06	23,48	16,88	44,46	50,34	21,36	9,10
5023	39,97	27,43	19,77	38,90	49,01	23,99	12,72
5031	92,63	56,56	36,31	33,15	54,22	20,45	10,19
5032	55,22	35,79	24,63	31,76	41,87	20,63	9,09
5041	37,14	27,17	21,77	42,11		25,86	10,75
5051		97,50	76,56	36,91		30,37	13,17
7511		25,30	27,84	75,73		64,16	32,80
7512		25,52	24,54	66,22		63,97	24,97
7513		25,69	24,34	72,63		50,16	21,96
7531		163,79	105,30	67,32		69,96	19,08
7532		145,55	104,39	95,31		77,09	22,96

Tabel 3.1

Literatuur

[1] Balinski, M.L. en Gomory, R.E., "A primal method for the assignment and transportation problems", Man. Sci. 10 (1964), 578-593.

[2] Dantzig, G.B., *Linear programming and extensions*, Princeton University Press, Princeton (1963).

[3] Dijkstra, E.W., "A note on two problems in connection with graphs", Num. Math. 1 (1959), 269-271.

[4] Domschke, W., *Kürzeste Wege in Graphen: Algorithmen, Verfahrensvergleiche*, Anton Hain, Meisenheim am Glan (1972).

[5] Dreyfus, S.E., "An appraisal of some shortest-path algorithms", Operations Res. 17 (1969), 395-412.

[6] Florian, M., en Klein, M., "An experimental evaluation of some methods of solving the assignment problem", CORS 8 (1970), 101-108.

[7] Ford, L.R., en Fulkerson, D.R., *Flows in Netwerks*, Princeton University Press, Princeton (1962).

[8] Geoffrion, A.U., en Marsten, R.E., "Integer Programming Algorithms: A Framework and State-of-the-Art-Survey", Man. Sci. 18 (1972), 465-491.

[9] Hillier, F.S., en Lieberman, G.J., *Introduction to Operations Research*, Holden-Day, San Francisco (1967).

[10] Klein, M., "A primal method for minimal cost flows with applications to the assignment and transportation problems", Man. Sci. 14 (1967), 205-220.

[11] Kuhn, H.W., "The Hungarian method for the assignment problem", Nav. Res. Log. Quart. 2 (1955), 83-97.

[12] Munkres, J., "Algorithms for the assignment and transportation problems", J. Soc. Industr. Appl. Math. 5 (1957), 32-38.

[13] Silver, R., "An Algorithm for the Assignment Problem", Comm. ACM, 3 (1960), 605-606.

[14] Silver, R., Algorithm 27, Assignment, Comm. ACM, 3 (1960), 603-604.

[15] Srinivasan, V., en Thompson, G.L., "Benefit-Cost Analysis of Coding Techniques for the Primal Transportation Algorithm", Journal of the ACM, 20 (1973), 194-213.

[16] Tabourier, Y., "Un algorithme pour le problème d'affectation", RAIRO 6 (1972), 3-15.

[17] Tomizawa, N., "On Some Techniques Useful for Solution of Transportation Network Problems", Networks 1 (1971), 173-194.

[18] Uebe, G., *Optimale Fahrpläne*, Springer, Berlin (1970).

[19] Votaw, D.F., "Methods of solving some personnel-classification problems", Psychometrica 17 (1952), 255-266.

[20] Votaw, D.F., en Orden, A., "The Personnel Assignment Problem", Project SCOOP, (1952), 155-163.

Appendix: ALGOL-procedure voor de verbeterde algoritme van Tomizawa

```
real procedure assignment(n,i,j,aij,x,u,v,eps); value n,eps;
integer n,i,j; real aij,eps; integer array x; array u,v;

begin integer i0,j0,k; real a,bg,dj,h,s,min,ui,vj;
    integer array y,lab[1:n]; real array d[1:n];

    bg:= 0;
    for j:= 1 step 1 until n do x[j]:= y[j]:= lab[j]:= 0;
    for i:= 1 step 1 until n do
    begin for j:= 1 step 1 until n do
        begin a:= aij; if bg<a then bg:=a;
            if if a+ui<0 then true else j=1 then
            begin ui:= -a; j0:= j end
        end;
        u[i]:= ui; if i=1 ∨ min+ui>0 then min:= -ui;
        if y[j0]=0 then begin y[j0]:= i; x[i]:= j0 end
    end;
    bg:= (bg-min)×n;

    for j:= 1 step 1 until n do
    v[j]:= if y[j]=0 then bg else 0;
    for i:= 1 step 1 until n do
    begin ui:= u[i];
        for j:= 1 step 1 until n do
        begin vj:= v[j]; if vj>eps then
            begin a:= aij+ui; if a<vj then
                begin v[j]:= a; lab[j]:= i end
            end
        end
    end;

    for j:= 1 step 1 until n do
    begin i:= lab[j]; if i=0 then else if x[i]=0 then
        begin x[i]:= j; y[j]:= i end
    end;

    for i:= 1 step 1 until n do if x[i]=0 then
    begin ui:= u[i];
        for j:= 1 step 1 until n do if y[j]=0 then
        begin a:= aij;
            if a+ui-v[j] < eps then
            begin x[i]:= j; y[j]:= i; goto l end
        end;
    l:
    end;
```

```
    for i0:= 1 step 1 until n do if x[i0]=0 then
    begin min:= bg; i:= i0;
        for j:= 1 step 1 until n do
        begin d[j]:= dj:= aij-v[j]; lab[j]:= -i0;
            if dj<min then begin min:= dj; j0:= j end
        end;

        u[i0]:= -min;
        if min=0 then else
        for j:= 1 step 1 until n do d[j]:= d[j]-min;
        i:= y[j0]; if i=0 then goto endkw; min:= 0;

        for k:= 1 step 1 until n do
        begin lab[j0]:= -lab[j0]; h:= min+u[i]; min:= bg;
            for j:= 1 step 1 until n do if lab[j]<0 then
            begin s:= h+aij-v[j]; dj:= d[j];
                if dj>s then begin d[j]:= dj:= s; lab[j]:= -i end;
                if dj<min then begin min:=dj; j0:= j end
            end;
            i:= y[j0]; if i=0 then goto endkw
        end;

    endkw: lab[j0]:= -lab[j0];
        for j:= 1 step 1 until n do
        begin h:= if lab[j]<0 then min else d[j];
            v[j]:= v[j]+h; i:= y[j]; if i>0 then u[i]:= u[i]+h
        end;

        for k:= j0 while k>0 do
        begin y[k]:= i:= lab[k]; j0:= x[i]; x[i]:= k end
    end;
    assignment:= sum(j,1,n,-u[j]+v[j])
end;
```