stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNDE BN 2/71 MEI

J.P. HOLLENBERG

EEN TWEE-DIMENSIONAAL VOORRAADPROBLEEM

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.

1. Omschrijving probleem

Een verkooporganisatie verkoopt twee bepaalde artikelen, vanaf nu te noemen produkt 1 en produkt 2. Deze artikelen kunnen besteld worden bij een andere organisatie, ze worden dan na $T > 0$ tijdseenheden geleverd en in een voorraadmagazijn opgeslagen. Er kan niet meer dan één order tegelijk uitstaan. Verder kan een bestelling op elk gewenst tijdstip gedaan worden. De inkoopkosten van d_1 eenheden van produkt 1 en d_2 eenheden van produkt 2 bedragen $\phi(d_1,d_2)$. Voor elke eenheid van produkt i zijn de voorraadkosten evenredig met de tijdsduur dat de eenheid in voorraad gehouden wordt, per tijdseenheid zijn de voorraadkosten $c_{1,i}$, $i = 1, 2$. Er kunnen maximaal M_i eenheden van produkt i in voorraad gehouden worden, $i = 1, 2$.

De klanten voor produkt i komen binnen volgens een Poisson proces met parameter λ_i; elke klant koop één eenheid, $i = 1, 2$. De beide Poisson processen zijn onderling onafhankelijk. Als een klant voor produkt i binnenkomt op een moment dat er geen voorraad van produkt i aanwezig is, dan wordt een noodinkoop verricht die $c_{2,i}$ kost, $i = 1, 2$.

Gevraagd wordt een inkoopstrategie te ontwerpen, zó dat de gemiddelde kosten per tijdseenheid in de long run minimaal zijn.

We zullen de oplossing iteratief bepalen met behulp van Markov-Programmering (zie syllabus 7c, §11 en §12).

2. Toestandruimte en toegelaten beslissingen

Wij nemen als toestandruimte

$$\mathcal{S} = \{(i_1,i_2) \mid 0 \leq i_r \leq M_r,\ r = 1, 2\} \cup$$

$$\{(i_1,i_2,d_1,d_2,t) \mid 0 \leq i_r \leq M_r,\ 0 \leq d_r \leq M_r - i_r,\ 0 \leq t < T,\ r=1,2\}.$$

De toestand $(i_1,i_2) = (i)$ correspondeert met de situatie dat de voorraad van produkt r de grootte i_r eenheden heeft voor r = 1, 2, en een bestelling gedaan kan worden indien gewenst. De toestand (i_1,i_2,d_1,d_2,t) correspondeert met de situatie dat voor r = 1, 2 geldt dat de voorraad van produkt r i_r eenheden bedraagt en er t tijdseenheden geleden een bestelling van d_r eenheden van dat produkt is gedaan.

Het natuurlijk proces verloopt als volgt. Als het natuurlijk proces (i) als begintoestand heeft dan blijft het systeem in toestand (i) tot het moment waarop een klant binnenkomt; op dat moment neemt het systeem een van beide toestanden $(\max(i_1-1,0),i_2)$ en $(i_1,\max(i_2-1,0))$ aan, afhankelijk van de vraag van de klant. Als het natuurlijk proces begint in toestand (i_1,i_2,d_1,d_2,t^*) met $0 \leq t^* < T$ dan neemt het systeem achtereenvolgens de toestanden (m_1,m_2,d_1,d_2,t) aan met $t^* < t < T$ en $m_r = \max[(i_r - \underline{v}_r(t-t^*)),0]$, r = 1, 2. Op het moment dat de levertijd T voorbij is neemt het systeem de toestand $(m_1^*+d_1, m_2^*+d_2)$ aan. Hierin is $m_r^* = \max[(i_r - \underline{v}_r(T-t^*)),0]$ en $\underline{v}_r$ de totale vraag in een tijdsinterval van de lengte t, voor r = 1, 2.

De beslisser kan alleen interveniëren in de toestanden (i) met $(i) < (M)$ *). Wij nemen aan dat in toestand (0,0) altijd een order geplaatst wordt. Als besloten wordt tot een bestelling van d_r eenheden van produkt r voor r = 1, 2 dan zullen we zeggen dat een beslissing (d) genomen wordt. De nulbeslissing geven we aan met (0) = (0,0). De verzameling

---

*) (i) = (j) wil zeggen $i_r = j_r$; r = 1, 2.
(i) $\leq$ (j) " " $i_r \leq j_r$; r = 1, 2.
(i) < (j) " " (i) $\leq$ (j) en niet (i) = (j).
(i) + (j) " " (i_1+j_1, i_2+j_2).
(i) $\neq$ (j) " " niet (i) = (j).

$\mathcal{A}(S)$ van toegelaten beslissingen in toestand S wordt gegeven door

$$\mathcal{A}(S) = \begin{cases} \{(d) \mid 1 \leq d_r \leq M_r \mid r = 1, 2\}, & S = (0), \\ \{(d) \mid 0 \leq d_r \leq M_r - i_r \mid r = 1, 2\}, & S = (i) \neq (0), \\ (0) & \text{elders.} \end{cases}$$

Door een interventie (d) in (i) gaat het systeem over in toestand $(i_1, i_2, d_1, d_2, 0)$.

Het zal duidelijk zijn dat we kiezen

$A_{01} = A_{02} = A_0 = \{(0,0)\}$ en derhalve geldt:

$\underline{w}_{10} = \underline{w}_{11} = \underline{w}_0$ en $\underline{w}_{20} = \underline{w}_{21} = \underline{w}_1$.

Een strategie z kent nu aan elke $s \in \mathcal{S}$ een $(d) \in \mathcal{A}(S)$ toe.

Op analoge wijze als in syllabus 7c voorbeeld 12.1 kan men nagaan dat wij ons voor het opstellen van de iteratiemethode kunnen beperken tot de toestanden (i).

3. Bepaling diverse overgangskansen

Om de k- en t-funkties te berekenen voeren wij de funkties $q^{(d)}_{(i)(j)}$ en $\beta((i)(j),A)$ in.

Definitie

$q^{(d)}_{(i)(j)}$ = de kans dat als er in (i) een bestelling (d) wordt gedaan die bestelling bij binnenkomst de toestand op (j) brengt, voor (i), (j) $\in \mathcal{S}$ en (d) $\neq$ (0), (d) $\in \mathcal{A}((i))$.

Wij definiëren eerst voor alle (i), (j) $\in \mathcal{S}$

$$q^{(0)}_{(i)(j)} = \begin{cases} 1 & \text{als } (i) = (j) \\ 0 & \text{elders,} \end{cases}$$

en verder omwille van de overzichtelijkheid van de notatie

$$\left.\begin{aligned} p_r(j) &= \begin{cases} e^{-\lambda_r T}\,\dfrac{(\lambda_r T)^j}{j!} & j \geq 0 \\ 0 & \text{elders} \end{cases} \\ P_r(j) &= \begin{cases} 1 - \sum_{k=0}^{j-1} p_r(k) & j \geq 0 \\ 1 & \text{elders} \end{cases} \end{aligned}\right\} \quad r = 1, 2.$$

Aangezien de klanten binnenkomen volgens Poissonprocessen zal het duidelijk zijn dat voor alle (i) < (M), alle (d) $\in \mathcal{A}(i)$ met (d) $\neq$ (0) geldt:

$$q^{(d)}_{(i)(j)} = \begin{cases} \prod_{r=1}^{2} p_r(i_r - j_r + d_r) & \text{als } i_r + d_r \geq j_r > d_r,\ r = 1, 2 \\ p_1(i_1 - j_1 + d_1).P_2(i_2) & \text{als } i_1 + d_1 \geq j_1 > d_1,\ j_2 = d_2 \\ P_1(i_1).p_2(i_2 - j_2 + d_2) & \text{als } j_1 = d_1,\ i_2 + d_2 \geq j_2 > d_2 \\ \prod_{r=1}^{2} P_r(i_r) & \text{als } j_r = d_r,\ r = 1, 2. \end{cases}$$

Zij A een verzameling met

$$\{(0)\} \subseteq A \subseteq \{(i) \mid 0 \leq i_r \leq M_r,\ r = 1, 2\}$$

dan komen wij tot de volgende definitie.

Definitie

$\beta((j)(k),A)$ = de kans dat een systeem met (j) als begintoestand en dat onderworpen is aan een natuurlijk proces als eerste toestand uit de verzameling A de toestand (k) zal aannemen.

Opmerking

De klanten voor produkt 1 resp. produkt 2 komen binnen volgens Poissonprocessen met parameters λ_1 resp. λ_2. Maken wij nu geen onderscheid tussen de typen klanten, dan kunnen wij stellen dat de klanten binnenkomen volgens een Poissonproces met parameter $\lambda_1 + \lambda_2$ en dat de kans dat een klant naar produkt i vraagt gegeven wordt door $\lambda_i/(\lambda_1+\lambda_2)$, i = 1, 2 (zie ook voorbeeld 12.2 in de syllabus 7c). Als we nu definieren

$$(i^*) = (i_1-1, i_2),\quad i_1 > 0,\quad (i) \in \mathcal{S},$$

$$(i^{**}) = (i_1, i_2-1),\quad i_2 > 0,\quad (i) \in \mathcal{S}$$

dan kunnen we $\beta((j)(k),A)$ als volgt rekursief berekenen

$$\beta((j)(k),A) = \begin{cases} 0 \quad \text{als niet geldt } (k) \leq (j) \\ 1 \quad \text{als } (j) \in A \text{ en } (k) = (j) \\ 0 \quad \text{als } (j) \in A \text{ en } (k) \neq (j) \\ \dfrac{\lambda_1}{\lambda_1+\lambda_2}\,\beta((j^*)(k),A) + \dfrac{\lambda_2}{\lambda_1+\lambda_2}\,\beta((j^{**})(k),A) \\ \qquad\qquad \text{als } (j) \notin A \text{ en } j_1, j_2 > 0 \\ \beta((j^*)(k),A) \quad \text{als } (j) = (j_1,0) \notin A \text{ en } j_1 \neq 0 \\ \beta((j^{**})(k),A) \quad \text{als } (j) = (0,j_2) \notin A \text{ en } j_2 \neq 0 \end{cases}$$

Definitie

$p^z_{(i)(j)}$ = de kans dat het systeem met begintoestand (i) en bij gevolgde strategie z de toestand (j) als eerste toestand van A_z aan zal nemen.

Er geldt

$$p^z_{(i)(j)} = \begin{cases} \beta((i)(j),A_z) & \text{als } (i) \notin A_z \\ q^{z(i)}_{(i)(j)} + \sum_{(k)\neq(j)} q^{z(i)}_{(i)(k)}\,\beta((k)(j),A_z) & \text{als } (i),(j) \in A_z \\ 0 & \text{elders.} \end{cases}$$

4. Berekening k- en t-funkties

Zoals boven reeds vermeld is kiezen we $A_0 = \{(0)\}$.

Uitgaande van een strategie z zullen we indien mogelijk verbeteren. Hiervoor zullen we de funkties k(i)(d) en t(i)(d) nodig hebben. We bepalen nu $t_0(i)$, $k_0(i)$, $t_1((i)(d))$ en $k_1((i)(d))$.

Definitie

$t_0(i)$ = de verwachting van de tijdsduur van $\underline{w}_0$ (= de stochastische wandeling van (i) naar A_0 als het systeem onderworpen is aan een natuurlijk proces).

Er geldt

$$t_0(i) = \begin{cases} i_1 / \lambda_1 & \text{als } i_1 \geq 0,\ i_2 = 0 \\ i_2 / \lambda_2 & \text{als } i_2 \geq 0,\ i_2 = 0 \\ \{1 + \lambda_1 t_0(i^*) + \lambda_2 t_0(i^{**})\} / (\lambda_1+\lambda_2) & \text{als } i_1,\ i_2 > 0 \end{cases}$$

Definitie

$t_1((i)(d))$ = de verwachting van de tijdsduur van $\underline{w}_1$ (= de stochastische wandeling van (i), waarin een order (d) geplaatst wordt, naar A_0 als het systeem onderworpen is aan een natuurlijk proces).

Er geldt voor alle $(i) \in \mathcal{S}$ en $(d) \in \mathcal{A}(i)$

$$t_1((i)(d)) = T.\delta(d_1+d_2) + \sum_{(j)\neq(i)}^{(i)+(d)} q_{(i)(j)}^{(d)}\, t_0(j),$$

waarin

$$\delta(u) = \begin{cases} 1 & \text{als } u > 0 \\ 0 & \text{elders.} \end{cases}$$

Definitie

$k_0(i)$ = de verwachting van de kosten te maken in de wandeling $\underline{w}_0$.

Het verwachte aantal klanten dat binnenkomt voor produkt r in een tijdsinterval van de lengte $t > 0$ is $\lambda_r t$, $r = 1, 2$. Het zal nu intuitief duidelijk zijn dat het verwachte aantal noodinkopen voor produkt r gedurende de wandeling $\underline{w}_0$ wordt gegeven door

$$\lambda_r t_0(i) - i_r \qquad r = 1, 2. \tag{1}$$

Een streng bewijs vindt men aan het eind van deze paragraaf.

Voor het bepalen van de voorraadkosten in $\underline{w}_0$ bedenken we dat de klanten voor produkt r binnen komen met een tussentijd die een verwachting $1/\lambda_r$ heeft. Hier volgt uit dat de verwachte voorraadkosten voor produkt r in de wandeling $\underline{w}_0$ zijn

$$\frac{c_{1,r}}{2\lambda_r}\, i_r(i_r+1).$$

We vinden nu

$$k_0(i) = \sum_{r=1}^{2} \{\frac{c_{1,r}}{2\lambda_r}\, i_r(i_r+1) + c_{2,r}(\lambda_r t_0(i) - i_r)\}.$$

Definitie

$k_1(i)(d)$ = de verwachting van de kosten te maken in $\underline{w}_1$.

We zullen de kosten splitsen in twee gedeeltes, n.l. die te maken als de bestelling (d) nog uitstaat en die daarna.

In de eerste periode (ter lengte T) zullen allereerst bestel- en inkoopkosten gemaakt worden, deze bedroegen volgens het gegeven $\phi(d)$. De noodinkoopkosten voor produkt r hebben een verwachting ter grootte van

$$c_{2,r} \sum_{k=i_r+1}^{\infty} (k-i_r)\, p_r(k), \qquad r = 1, 2.$$

Door een redenering analoog aan die welke in voorbeeld 10.4 uit syllabus 7c gevolgd wordt om de voorraadkosten te berekenen, kunnen we stellen dat de voorraadkosten voor produkt r de verwachting hebben van

$$\frac{c_{1,r}}{2\lambda_r}\{(i_r+1)\, i_r - \sum_{m=1}^{i_r} p_r(i_r-m)\, m(m+1)\}, \quad r = 1, 2.$$

De verwachting van de kosten te maken na het binnenkomen van de order in de wandeling $\underline{w}_1$ is

$$\sum_{(j)=(d)}^{(i)+(d)} q^{(d)}_{(i)(j)}\, k_0(j).$$

Een en ander geeft als resultaat

$$k_1(i) = \phi(d) + \sum_{(j)=(d)}^{(i)+(d)} q^{(d)}_{(i)(j)}\, k_0(j) +$$
$$+ \sum_{r=1}^{2} [c_{2,r} \sum_{k=i_r+1}^{\infty} (k-i_r)\, p_r(k) + \frac{c_{1,r}}{2\lambda_r}\{(i_r+1)\, i_r +$$
$$- \sum_{m=1}^{i_r} p_r(i_r-m)\, m(m+1)\}].$$

Wij merken op dat (zie appendix uit deel 7c)

$$\sum_{k=i_r+1}^{\infty} (k-i_r)\, p_r(k) = \lambda_r T\, P_r(i_r-1) - i_r\, P_r(i_r).$$

Verder geldt

$$k((i)(d)) = k_1((i)(d)) - k_0(i)$$

en

$$t((i)(d)) = t_1((i)(d)) - t_0(i),$$

wij merken verder op dat als (d) = (0)

$$k((i)(d)) = t((i)(d)) = 0.$$

Wij zullen nu formule (1) op blz. 8 bewijzen. Wij doen dit voor $r = 1$, het geval $r = 2$ gaat analoog.

Laat $V_1(i)$ de verwachting van de vraag zijn naar produkt 1 in de wandeling $\underline{w}_0$ met begintoestand (i). Uiteraard geldt

$$V_1(i) = \lambda_1 \, t_0(i) \qquad \text{voor } (i) = (i_1,0) \qquad (2)$$

omdat

$$t_0(i) = i_1/\lambda_1 \qquad \text{voor } (i) = (i_1,0).$$

Omdat de klanten voor de produkten 1 en 2 binnenkomen volgens onafhankelijke Poissonprocessen met parameters λ_1 resp. λ_2, geldt

$$V_1(i) = \lambda_1 \, t_0(i) \qquad \text{als } (i) = (0,i_2). \qquad (3)$$

Neem nu $(i) = (i_1,i_2) \geq (1,1)$, dan geldt

$$t_0(i) = \frac{1}{\lambda_1+\lambda_2} + \frac{\lambda_1}{\lambda_1+\lambda_2} t_0(i^*) + \frac{\lambda_2}{\lambda_1+\lambda_2} t_0(i^{**}) \qquad (4)$$

en

$$V_0(i) = (1+V_0(i^*)) \frac{\lambda_1}{\lambda_1+\lambda_2} + V_0(i^{**}) \frac{\lambda_2}{\lambda_1+\lambda_2} \quad . \qquad (5)$$

Door substitutie volgt uit (2) t/m (5) dat

$$V_0(i) = \lambda_1 \, t_0(i).$$

Hieruit volgt nu relatie (1).

5. Iteratie stap

Uitgaande van strategie z construeren we een nieuwe strategie die indien mogelijk beter is. We gebruiken de volgende methode:

Bepaal de priemfuiken $K_1, \ldots, K_m$ van A_z en kies in K_j een toestand $(i)_j$, $1 \leq j \leq m$.

Bepaal de unieke oplossing $y((i),z)$ en $v((i)z)$ uit

$$\begin{cases} y(i) = \sum\limits_{(j)\in A_z} y(j)\, P^z_{(i)(j)} & (i) \in A_z \\ v(i) = k(i)z(i) - y(i)t(i)z(i) + \sum\limits_{(j)\in A_z} v(j) p^z_{(i)(j)} & (i) \in A_z \\ v(i) = 0 \qquad (i) = (i)_j & j = 1, \ldots, m. \end{cases}$$

Bereken vervolgens

$$y((i),z) = \sum_{(j)\in A_z} y((j),z)\, p^z_{(i)(j)} \qquad (i) \notin A_z$$

$$v((i),z) = \sum_{(j)\in A_z} v((j),z)\, p^z_{(i)(j)} \qquad (i) \notin A_z,$$

wat neerkomt op $((i) \notin A_z)$

$$v((i),z) = \begin{cases} (\lambda_1 v((i^*),z) + \lambda_2 v((i^{**}),z))/(\lambda_1+\lambda_2) & i_1, i_2 > 0 \\ v((i^*),z) & i_1 > 0,\ i_2 = 0 \\ v((i^{**}),z) & i_1 = 0,\ i_2 > 0 \end{cases}$$

en $y((i),z)$ kan evenzo rekursief berekend worden.

Analoog als in voorbeeld 12.1 vinden we de testgrootheden

$$\hat{v}_z((i)(d)) = k(i)(d) - y((i),z)t(i)(d) + \sum_{(j)=(d)}^{(i)+(d)} v((i),z) q^{(d)}_{(i)(j)}$$

$$\hat{y}_z((i)(d)) = \sum_{(j)=(d)}^{(i)+(d)} y((i),z) q^{(d)}_{(i)(j)} \quad .$$

Bepaal voor elke (i) $\in \mathcal{S}$ de verzameling $\mathcal{X}_1(i)$ bestaande uit die beslissingen voor welke $\hat{y}_z((i)(d))$ onder (d) $\in \mathcal{X}((i))$ minimaal is, noem dat minimum $\hat{y}((i),z)$.

Voeg aan elke (i) een beslissing $(d)_{(i)}$ uit $\mathcal{X}_1(i)$ toe waarvoor $\hat{v}_z((i)(d))$ onder (d) $\in \mathcal{X}_1(i)$ minimaal is. Wordt dat minimum voor meerdere beslissingen bereikt dan heeft z(i) de voorkeur, mits natuurlijk $\hat{v}_z((i)z(i))$ gelijk is aan het minimum. Dat minimum noemen we $\hat{v}((i),z)$.

Definieer z' als volgt: $z'(i) = (d)_{(i)}$.

Construeer de kleinste verzameling A, waarvoor geldt

$$A_0 \subseteq A \subseteq A_{z'},$$

én

$$\begin{cases} \hat{y}((i),z,A) > \hat{y}((i),z) & (i) \in A_{z'}, \\ \text{óf} \\ \begin{cases} \hat{y}((i),z,A) = \hat{y}((i),z) & (i) \in A_{z'}, \\ \text{én} \\ \hat{v}((i),z,A) \geq \hat{v}((i),z) & (i) \in A_{z'}, \end{cases} \end{cases}$$

waarbij de volgende definities gelden.

Definities

$$\hat{y}((i),z,A) = E\hat{y}((\underline{i}),z)$$

en

$$\hat{v}((i),z,A) = E\hat{v}((\underline{i}),z)$$

waarbij $(\underline{i})$ de eerst aangenomen toestand van A is als de begintoestand (i) is en het systeem onderworpen is aan een natuurlijk proces.

Laat nu $A'_{z'}$ de kleinste verzameling A zijn met deze eigenschap. Definieer nu de strategie z" als volgt

$$z''(i) = \begin{cases} z'(i) & \text{als } (i) \in A'_{z'}, \\ (0) & \text{elders.} \end{cases}$$

De bedoeling van de constructie van A was een uiteindelijke strategie te krijgen door aan z' wat nulbeslissingen toe te voegen (een stukje van A_z, af te snijden). We zullen zien hoe het een en ander in de praktijk (meestal) toegaat.

In de praktijk komen priemfuiken zelden voor. Indien we aannemen dat ze nooit voorkomen dan is

$$y((i),z) = \hat{y}_z(i)(d) = \hat{y}((i),z) = \hat{y}((i),z)_A = y(z) \quad \forall (i) \in \mathcal{S}$$

en dus is

$$\mathcal{H}_1(i) = \mathcal{H}(i) \qquad \forall (i) \in \mathcal{S}.$$

We construeren z_1 verder als boven beschreven en testen dan voor elke toestand in A_z, of die wel in A_z, zal blijven. Van (0,0) is dat zeker en we testen de toestanden in zo'n volgorde dat als we een toestand (j) testen alle toestanden (i) met (i) < (j) reeds getest zijn. We kunnen die gewenste volgorde van toestanden bereiken door b.v. de volgorde $(0,1), \ldots, (0,M_2), (1,0), \ldots, (1,M_2), \ldots, (M_1-1,M_2), (M_1,0), \ldots, (M)$ aan te houden.

We bedenken dat

$$\hat{v}((i),z,A) = E\hat{v}((\underline{i}),z) = \sum_{(j)\in A} \beta((i)(j),A).\hat{v}((i),z)$$

en voeren de hulpgrootheden w(i) en $\hat{w}(i)$ in. We definieren

$$w(0) = \hat{v}((0),z)$$

en gaan daarna in de boven beschreven volgorde voor elke toestand als volgt te werk:
Als $(i) \notin A_z$, definieer dan

$$w(i) = \begin{cases} (\lambda_1 w(i^*) + \lambda_2 w(i^{**}))/(\lambda_1+\lambda_2) & \text{als } i_1, i_2 > 0 \\ w(i^*) & \text{als } i_1 > 0, i_2 = 0 \\ w(i^{**}) & \text{als } i_1 = 0, i_2 > 0, \end{cases}$$

als $(i) \in A_z$, definieer dan

$$\hat{w}(i) = \begin{cases} (\lambda_1 w(i^*) + \lambda_2 w(i^{**}))/(\lambda_1+\lambda_2) & \text{als } i_1, i_2 > 0 \\ w(i^*) & \text{als } i_1 > 0, i_2 = 0 \\ w(i^{**}) & \text{als } i_1 = 0, i_2 > 0, \end{cases}$$

en test dan vervolgens of er geldt

$$\hat{w}(i) > \hat{v}((i),z).$$

Als dat het geval is, dan definiëren wij

$$w(i) = \hat{v}((i),z)$$

en indien $\hat{w}(i) \leq \hat{v}((i),z)$ dan definiëren wij

$$w(i) = \hat{w}(i)$$

en we verwijderen (i) uit A_z,.

6. Voorbeeld

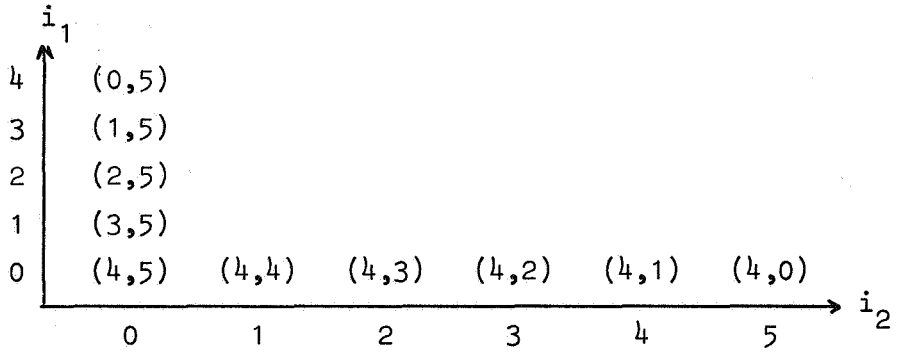
Data:

	r=1	r=2
M_r	4	5
λ_r	1	2
$c_{1,r}$	2	3
$c_{2,r}$	16	17

en verder geldt: T = 1 en

$$Q(d_1,d_2) = 2d_1 + 3d_2 + \delta(d_1) + \delta(d_2) + \delta(d_1+d_2) + \delta(d_1d_2).$$

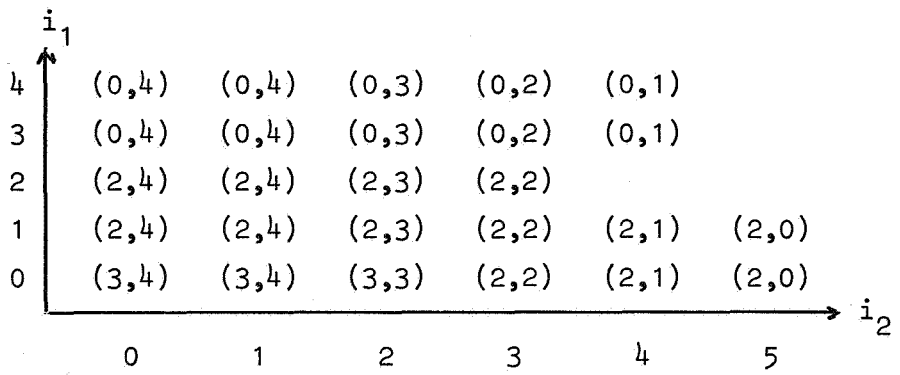
Begin strategie.



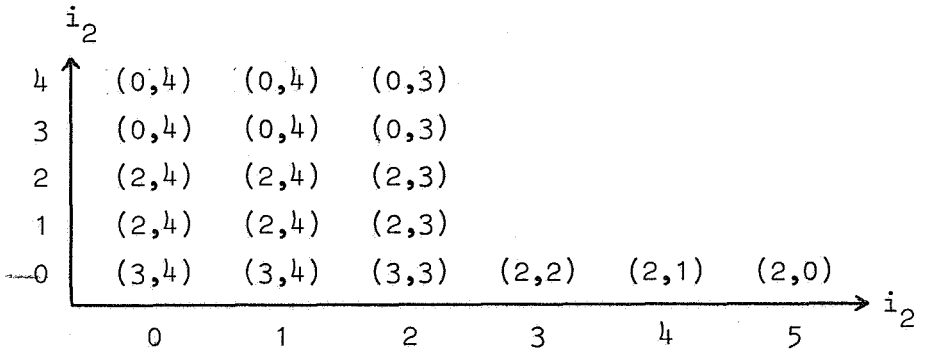
y = 29.60.

Eerste stap.

na policy improvement operation:



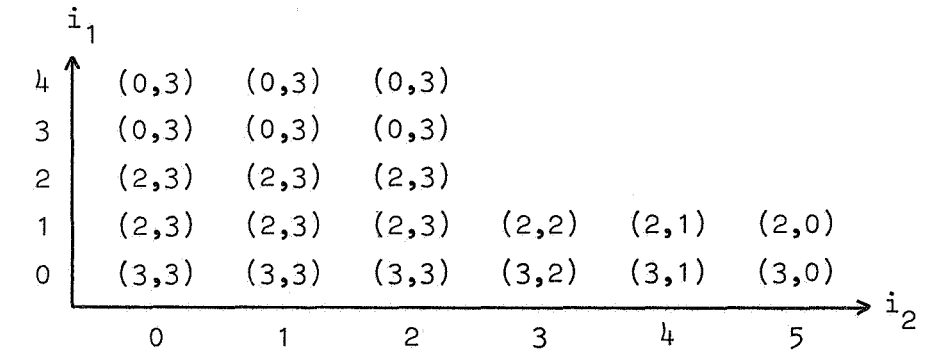
na afsnijstap:



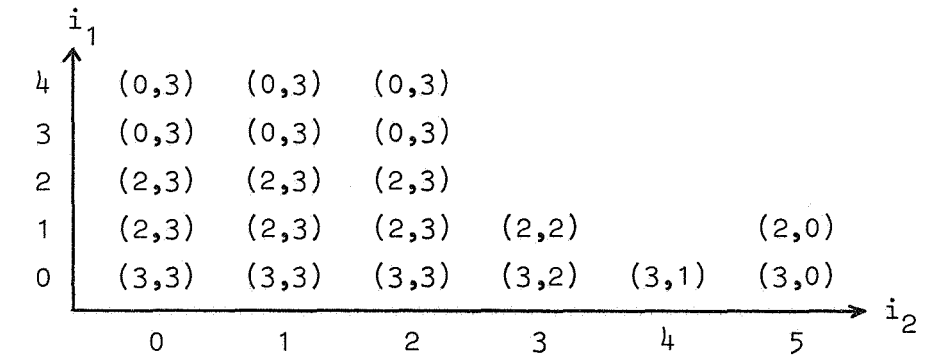
y = 26.64.

Tweede stap.

na policy improvement operation:



na de afsnijstap:



y = 26.45.

Dit is tevens de eindstrategie want de derde stap levert hetzelfde resultaat.

De eindstrategie z geeft de volgende waarden van v((i),z):

i_1 \ i_2	0	1	2	3	4	5
4	-66.27	-64.70	-60.09	-53.21	-47.19	-42.18
3	-50.08	-48.75	-44.84	-39.43	-35.15	-32.16
2	-34.02	-33.62	-31.56	-28.60	-26.60	-26.18
1	-19.00	-21.20	-22.60	-22.69	-22.61	-25.35
0	-.00	-10.08	-16.76	-20.37	-22.44	-26.92

7. Opmerkingen van programma-technische aard

We gaan uit van strategie z met A_z die k toestanden bevat, verder nemen we gemakshalve n = M_1+1 en m = M_2+1. Duidelijk is $0 \leq k < nm-1$.

Zonder discussie kunnen we stellen dat het de voordeligste methode is de benodigde funktiewaarden van $p_r(j)$, $P_r(j)$, $k_0(i)$ en $t_0(i)$ reeds voor de eerste iteratiestap in het geheugen op te slaan. In de iteratie stap moeten we o.a. een vierkant stelsel van k+1 vergelijkingen met k+1 onbekenden oplossen. Aangezien we in het algemeen de waarde van k niet kunnen voorspellen zullen we reeds voor de eerste stap een 2-zijdig a. array moeten reserveren (d.w.z. $(mn)^2.2$ geheugenplaatsen). Het oplossen van het stelsel vergt een aantal bewerkingen dat evenredig is met $(k+1)^3$.

Voor de waarden van $p^z_{(i)(j)}$ in de iteratie stap hebben we de keuze tussen

a) rekursieve procedure aanroepen,

b) de waarden van $p^z_{(i)(j)}$ in een array opslaan.

Aangezien mogelijkheid a) hier veel overbodig werk en geheugenbezetting zou vergen verdient mogelijkheid b) hier de voorkeur. Dit vergt $2(nm)^2$ geheugenplaatsen doch geen extra ruimte daar we de gereserveerde plaats voor de linkerleden van de k+1 vergelijkingen (zie boven) daarvoor "even" kunnen gebruiken.

Tijden het uitvoeren van de policy improvement operation moeten we $\hat{v}_z((i)(d))$ n(n+1) m(m+1)/4 keer berekenen. Hierbij is een tijdrovende omstandigheid dat $\hat{v}_z((i)(d))$ de term $\sum_{(j)=(d)}^{(i)+(d)} v((i),z)\, q^{(d)}_{(i)(j)}$ bevat.
Een soortgelijk kenmerk hebben de k- en t-funkties ook.
Voor deze drie funkties hebben we de keuze (C_1 en C_2 zijn konstanten):

a) de funktie elke keer als hij nodig is berekenen dat komt neer op $C_1(nm)^3$ berekeningen voor $\hat{v}_z((i)(d))$ en $C_2((nm)^3+k)$ berekeningen voor de k- en t-funkties.

b) de funkties of delen daarvan vooruit te berekenen (vóór de eerste stap wat betreft de k- en t-funkties en vóór elke stap wat betreft $\hat{v}_z((i)(d))$). Dit komt neer op $C_2(nm)^2$ bewerkingen vóór de eerste stap en $C_1(nm)^2$ bewerkingen vóór elke stap. Bovendien vergt dit minmaal $3.(nm)^2$ geheugenplaatsen.

Voor kleine nm kunnen we het beste $\hat{v}$ volgens a) bepalen en k en t volgens b). Deze laatste 2 funkties kunnen vóór de eerste stap in één array opgeslagen worden en nemen zo $2(nm)^2$ plaatsen in beslag.
Voor grote nm zal zowel a) als b) niet voldoen wegens óf te lange rekentijd óf te grote vraag naar geheugenruimte.

8. Input

De kostenfunktie $\phi(d)$ hebben wij gesteld op

$$c_{3,1}d_1 + c_{3,2}d_2 + c_{4,1}\delta(d_1) + c_{4,2}\delta(d_2) + c_5\delta(d_1+d_2) + c_6\delta(d_1d_2).$$

De hierin voorkomende konstante coëfficiënten worden met de andere konstanten ingelezen in de onderstaande volgorde:

M_1 M_2	(maximum voorraad)
λ_1 λ_2	(parameters van de Poissonprocessen)
$C_{1,1}$ $C_{1,2}$	(voorraadkosten)
$C_{2,1}$ $C_{2,2}$	(noodinkoopkosten)
T	(levertijd)
$C_{3,1}$ $C_{3,2}$	(konstanten van $\phi(d)$)
$C_{4,1}$ $C_{4,2}$	
C_5	
C_6	
i	(aanduiding van de precisie)

Indien we voor c_6 een negatief getal inlezen dan neemt het programma voor c_6 een getal in de orde van grootte van 10^{629}. Op deze manier kunnen we inkopen van beide produkten tegelijk "verbieden". De laatste konstante is een getal i (i geheel en positief). Deze gebruiken we in de policy improvement operation, waar we $\hat{v}((i),z)$ bepalen. Hier worden twee waarden van $\hat{v}_z((i)(d))$ gelijkgesteld als voor hun quotient Q geldt dat $|1-Q| < 10^{-i}$.

Zodra het programma een probleem behandeld heeft begint het aan een nieuw probleem, dit houdt op als in plaats van een nieuwe serie data een negatief getal ingelezen wordt.