EGY SZTOHASZTIKUS PROGRAMOZÁSI MODELL SZÁMITÓGÉPES KIÉRTÉKELÉSE*

Deák István

A cikkben egy Prékopa András által megfogalmazott és megoldott sztohasztikus programozási probléma számítógépes, numerikus kiértékelését adjuk. Valamely matematikai feladatot megoldó algoritmus praktikussága felől az algoritmus számítógépes reprezentációja és megfelelő számú kísérleti feladat megoldása után mondhatunk csak véleményt. Különösen igaz ez a sztohasztikus programozási modelleket megoldó algoritmusok esetében. Az elméleti algoritmusok ismeretében sem a konvergenciasebességről, sem a kiszámításhoz szükséges időről nem tudunk konkrétumot mondani, továbbá egy elméleti algoritmus még egyáltalán nem számol a konkrét helyzet számítástechnikai apparátusával. Ezért szükséges, hogy a programozási modellek számítógépes programját is elkészítsük, ezzel a gyakorlatban oldjuk meg a felmerülő számítástechnikai problémákat. Több konkrét feladat lefuttatásával a feladat kiszámításához szükséges időről is jó becsléseket kaphatunk.

A jelen dolgozat részben a szerző egyetemi doktori disszertációjában foglaltakat, részben azok továbbfejlesztését tartalmazza. Az első részben a sztohasztikus programozási modellt és az optimális megoldást kiszámító algoritmust írjuk le, a második részben a felmerült számítástechnikai nehézségeket és megoldásukat, a harmadik részben pedig konkrét példák futtatásával kapcsolatos tapasztalatokat közlünk.

A gépi programot először ALGOL nyelven az Egyetemi Számítóközpont Razdan 3 gépére, majd szintén ALGOL nyelven az Akadémia Számítástechnikai Központjának CDC 3300-as gépére készítettem el. A legújabb változat pedig FORTRAN nyelven a CDC 3300-as gépen működik.

1. A SZTOCHASZTIKUS PROGRAMOZÁSI MODELL ÉS AZ OPTIMÁLIS MEGOLDÁST KISZÁMITÓ ALGORITMUS

Tekinsük a következő determinisztikus modellt:

$$
\begin{aligned}
& g_i(\underline{x}) \geqslant \beta_i\,, \qquad i = 1, \ldots, n\,, \\
& \underline{a}_i' \underline{x} \geqslant b_i\,, \qquad i = 1, \ldots, M\,, \\
& \min f(\underline{x})\,.
\end{aligned}
\tag{1.1}
$$

A $g_i(\underline{x})$ függvények konkávak, az $f(\underline{x})$ függvény konvex, a β_i, b_i-k valós számok, $\underline{a}_i$-k pedig oszlopvektorok. Tegyük most fel, hogy β_i-k valószínűségi változók, akkor az eredeti (1.1)

*Ez a munka az Országos Tervhivatal Tervgazdasági Intézetének támogatásával készült.

probléma értelmét veszti, fogalmazzuk meg a következő sztohasztikus programozási feladatot:

$$
\begin{aligned}
& P\{g_i(\underline{x}) \geqslant \beta_i\,, \qquad i = 1, \ldots, n\} \geqslant p\,, \qquad 0 < p < 1\,, \\
(1.2) \qquad & \underline{a}_i' \underline{x} \geqslant b_i\,, \qquad i = 1, \ldots, M\,, \\
& \min f(\underline{x})\,.
\end{aligned}
$$

Ezt a modellt Prékopa András [4] állította fel, és ő is adta meg bizonyos, itt nem részletezendő feltételek mellett az optimális megoldáshoz vezető algoritmust. Maga az algoritmus ismert volt (Zoutendijk [5]), de Prékopa bizonyította be, hogy ebben az esetben is konvergál.

Vezessük be a következő jelöléseket. Tegyük fel, hogy a β_i-k együttes eloszlása normális, várható értékük zérus, szórásuk egy, – ellenkező esetben a várható értékeket levonva és a szórással végigosztva a megfelelő feltételi egyenlőtlenséget kaphatjuk ezt az alakot – korrelációs mátrixuk R. Tehát a β_i-k együttes eloszlásfüggvénye:

$$
\Phi(y_1, \ldots, y_n) = \frac{1}{(2\pi)^{\frac{n}{2}} \cdot \sqrt{|R|}} \int_{-\infty}^{y_n} \ldots \int_{-\infty}^{y_1} \exp\{-\frac{1}{2} \underline{x} R^{-1} \underline{x}\} dx\,.
$$

Ha a β_i-edik együttes eloszlása normális, akkor teljesülnek az algoritmus konvergenciájához szükséges feltételek. Jelöljük:

$$
G(\underline{x}) = \Phi(g_1(\underline{x})_1, \ldots, g_n(\underline{x})) = P\{g_1(\underline{x}) \geqslant \beta_1, \ldots, g_n(\underline{x}) \geqslant \beta_n\}\,.
$$

Ezekkel a jelölésekkel az (1.2) probléma a következőképpen írható fel:

$$
\begin{aligned}
& G(\underline{x}) \geqslant p\,, \qquad 0 < p < 1\,, \\
(1.3) \qquad & \underline{a}_i' \underline{x} \geqslant b_i\,, \qquad i = 1, \ldots, M, \\
& \min f(\underline{x})\,.
\end{aligned}
$$

Szükségünk van egy $\underline{x}_1$-re, amely megengedett megoldása (1.3)-nak. Ebből kiindulva további $\underline{x}_2, \ldots, \underline{x}_k, \ldots,$ megengedett megoldásokat készítünk az alábbi módon.

Két eset lehetséges, vagy ez a sorozat véges, ebben az esetben valamely k-ra $\underline{x}_k$ optimális megoldása lesz (1.3)-nak, vagy a megengedett megoldások így képzett sorozata végtelen és ekkor konvergál az (1.3) optimális megoldásához.

Tegyük fel, hogy már van egy $\underline{x}_k$ vektorunk, amely megengedett megoldása (1.3)-nak, ekkor $\underline{x}_{k+1}$-et a következőképpen kaphatjuk. Linearizáljuk az (1.3) első feltételét az $\underline{x}_k$ pontban vett Taylor sorával, vegyünk hozzá még egy sort, amely a célfüggvényt veszi figyelembe, így felírhatjuk a következő lineáris feladatot:

$$
\begin{aligned}
& G(\underline{x}) + \nabla G(\underline{x}_k)(\underline{x} - \underline{x}_k) + \vartheta y \geqslant p\,, \qquad 0 < p < 1, \\
& \underline{a}_i' \underline{x} \geqslant b_i\,, \qquad i = 1, \ldots, M\,, \\
& \nabla f(\underline{x}_k)(\underline{x} - \underline{x}_k) \leqslant y\,, \\
& \min y\,.
\end{aligned} \tag{1.4}
$$

Itt $\vartheta > 0$ rögzített szám az egész eljárás folyamán, egyébként tetszőleges. Bizonyos, elég enyhe feltételek mellett ennek a feladatnak létezik optimális megoldása [4], legyen ez (x_k^*, y_{opt}). Látható, hogy $(\underline{x}_k, 0)$ megengedett megoldása (1.4)-nek. Ha ez optimális megoldása is (1.4)-nek, vagyis $\underline{x}_k^* = \underline{x}_k$, $y_{\text{opt}} = 0$, akkor ez optimális megoldása (1.3)-nak is. A fentebb tárgyalt két eset közül az elsővel állunk ilyenkor szemben, $\underline{x}_k$ optimális megoldása az eredeti sztohasztikus problémának, végetér az algoritmus. Ha viszont $y_{\text{opt}} \neq 0$, akkor új megengedett megoldását készítjük el (1.3)-nak.

Tekintsük a következő félegyenest:

$$\underline{x}_k + \lambda(\underline{x}_k^* - \underline{x}_k), \qquad \lambda > 0\,. \tag{1.5}$$

Minimalizáljuk $f(\underline{x})$-t ezen félegyenesnek az (1.3) megengedett megoldásainak halmazába eső részén.

Vagyis legyen μ_k az a legnagyobb λ, amelyre még igaz

$$
\begin{aligned}
& G(\underline{x}_k + \lambda(\underline{x}_k^* - \underline{x}_k)) \geqslant p\,, \qquad 0 < p < 1\,, \\
& \underline{a}_i'(\underline{x}_k + \lambda(\underline{x}_k^* - \underline{x}_k)) \geqslant b_i\,, \qquad i = 1, \ldots, M\,,
\end{aligned}
$$

és legyen λ_k egy olyan λ, amelyre

$$f(\underline{x}_k + \lambda(\underline{x}_k^* - \underline{x}_k)) \geqslant f(\underline{x}_k + \lambda_k(\underline{x}_k^* - \underline{x}_k)), \qquad 0 \leqslant \lambda < \mu_k\,.$$

Ekkor definiáljuk $\underline{x}_{k+1}$-et a következőképpen:

$$\underline{x}_{k+1} = \underline{x}_k + \lambda_k(\underline{x}_k^* - \underline{x}_k)\,. \tag{1.6}$$

A fentiek szerint ez megengedett megoldása (1.3)-nak. Ezzel készítsük el az új (1.4) lineáris problémát és ismételjük meg az egész eljárást. Igy kapjuk az $\underline{x}_1, \ldots, \underline{x}_k, \ldots$ megengedett megoldások véges vagy végtelen sorozatát, amelynek utolsó tagja az (1.3) optimális megoldása lesz, illetőleg a sorozat konvergál az optimális megoldáshoz.

2. SZÁMITÁSTECHNIKAI PROBLÉMÁK ÉS MEGOLDÁSUK

A gyakorlatban előforduló problémáknál $n = 10 - 15$, $M = 100 - 200$ szokott lenni. Ezért a programba beépített számítástechnikai eljárásokat igyekeztem úgy megválasztani, hogy azok ne csak alacsony $(n = 2, 3)$ dimenziószám esetén működjenek jól, hanem magasabb

dimenzióban is egy reális gépi idő alatt lefussanak. A készített program elvileg akármilyen nagy feladatot tud kezelni, gyakorlatilag a következő méretű feladat futtatható le: $n = 5$, $M = 30$, $\underline{x}$ vektor 30 dimenziós.

Az elkészített program és a lefuttatott feladatok arra az esetre vonatkoznak, ha a $g_i(\underline{x})$ és $f(\underline{x})$ függvények lineárisak, a β_i-k együttes eloszlása normális, a feladat méretei $n = 2$, $M = 2$ és az $\underline{x}$ vektor is két dimenziós.

A legnagyobb problémát a többdimenziós normális eloszlásfüggvény adott pontban felvett értékének gyors és lehetőleg pontos kiszámítása jelentette. Ez azért is fontos volt, mert a modell első gépi programjában a futási idő 90 %-ában a gép a normális eloszlásfüggvény egyes értékeit számította. Problémák léptek fel az (1.5) félegyenesen való haladás közben, annak megállapításával kapcsolatban, hogy mikor érjük el a megengedett megoldások tartományának határát.

Nehézséget okozott még a közelítés jóságának megállapítása, vagyis az iterálás befejezésének feltétele. Végül legjobbnak az kínálkozott, ha több kritérium együttes teljesülése esetén állítjuk le az iterálást.

A továbbiakban ezt a három problémakört fogjuk külön-külön megvizsgálni.

a/ A többdimenziós normális eloszlásfüggvény adott pontban felvett értékének kiszámítása.

Táblázatok vagy közelítő képletek csak igen speciális korrelációs mátrixok esetén és csak alacsony dimenzióban ($n = 2, 3, 4$) léteznek [3]. A felhasznált számítástechnikai eljárást viszont úgy kellett kiválasztani, hogy $n = 10 - 15$ dimenzióban is hatékony legyen. Ez a szempont eleve kizárt minden hagyományos megközelítést (táblázatok, közelítő képletek, stb.). Egyetlen járható útnak a Monte Carlo módszerek mutatkoztak. Ezek között lényegében két eljárás kínálkozott a többdimenziós normális eloszlásfüggvény adott pontbeli értékének kiszámítására. Tegyük fel, hogy a

$$p = \Phi(h_1, \ldots, h_n)$$

értéket akarjuk kiszámítani. Legyen D a következő:

$$D = \{ \underline{y} : \underline{y} \leqslant \underline{h} \} .$$

Ekkor nyilván

$$P = P\{\chi \in D\} , \tag{2.1}$$

ahol χ egy normális eloszlású valószínűségi vektorváltozó. A gyakorlati kiszámítás során D helyett a

$$D^* = \{ \underline{y} : -\underline{a} \leqslant \underline{y} \leqslant \underline{h} \}$$

tartományt használjuk, ahová az $\underline{a} = \underline{4,5}$ vektort helyettesítjük. A tartomány ilyen csonkolása által elkövetett hiba elhanyagolhatóan kicsi a Monte Carlo módszerek hibáihoz képest, ha a szórása minden komponensnek 1, viszont ez a csonkolás a számításokat egyszerűbbé teszi. A (2.1) szerint tehát ha N db R korrelációs mátrixú normális eloszlású pszeudóvéletlen vektorváltozót generálok és ha N_1 esik ezek közül a D tartományba, akkor az $\frac{N_1}{N}$ relatív gyakoriság értéke a p érték egy becslése lesz. A másik módszer a p érték meghatározására az

$$p = \int_D \ldots \int \varphi(\underline{y})\, d\underline{y} \tag{2.2}$$

integrál közelítő kiszámítása valamely Monte Carlo módszerrel, ahol $\varphi(\underline{y})$ a többdimenziós normális eloszlású sűrűségfüggvénye.

I. *A relatív gyakoriság kiszámítása.*

A feladat N darab $\underline{\chi}^{(i)}$ pszeudovéletlen R korrelációmátrixú normális eloszlású vektorváltozó generálása. Képezzük először a

$$\xi = \sqrt{\frac{12}{k}} \left[(\zeta_1 + \ldots + \zeta_k) - \frac{k}{2} \right] \tag{2.3}$$

összeget, ahol ζ_i a (0, 1]-ben egyenletes eloszlású pszeudovéletlen szám [1]. Az összeg eloszlása Ljapunov tétele szerint $k \to \infty$ esetén az $N(0, 1)$ standard normális eloszláshoz konvergál. Rögtön látható, hogy $M(\xi) = 0$ és $D(\xi) = 1$. Ha az így kapott ξ-ket egy $\underline{\eta}$ n dimenziós vektor egyes komponenseinek helyébe vesszük, akkor $\underline{\eta}$ már egy n dimenziós pszeudovéletlen normális eloszlású vektorváltozó, amelynek komponensei függetlenek. Ezt az $\underline{\eta}$ vektort egy $\underline{\chi}$ n dimenziós C korrelációs mátrixú normális eloszlású vektorváltozóvá legcélszerűbben egy A háromszögmátrixszal lehet transzformálni.

Vagyis legyen

$$\begin{aligned} \chi_1 &= a_{11} \cdot \eta_1 , \\ \chi_2 &= a_{21} \cdot \eta_1 + a_{22}\eta_2 , \\ &\cdot \\ &\cdot \\ &\cdot \\ \chi_n &= a_{n1} \cdot \eta_1 + a_{n2} \cdot \eta_2 + \ldots + a_{nn} \cdot \eta_n . \end{aligned} \tag{2.4}$$

Az A mátrix a_{ij} elemét úgy határozzuk meg, hogy $M(\chi_k \chi_l) = r_{kl}$ legyen, ahol r_{kl} az R korrelációs mátrix eleme. Az A mátrix elemeinek kiszámítása után könnyen származtathatjuk a keresett $\underline{\chi}^{(1)}, \ldots, \underline{\chi}^{(N)}$ vektorokat:

$$\underline{\chi}^{(i)} = A \cdot \underline{\eta}^{(i)} . \tag{2.5}$$

Megjegyezzük még, hogy a (2.3) összeg konvergenciáját speciális transzformációkkal gyorsítani lehet [2]. Vegyük a

$$\xi' = \frac{1}{\sqrt{k}} \sum_{i=1}^{k}{}' \zeta_i$$

összeg segítségével képzett

$$\xi = \xi' - \frac{1}{20k}(3\xi' - \xi'^3)$$

valószínűségi változót a (2.3) összeg helyett, ahol ζ_i (0, 1]-ben egyenletes eloszlású szám. Az így képzett ξ pszeudovéletlen szám empirikus eloszlása jobban megközelíti (adott k esetén) a normális eloszlást, mint a (2.3) alatti összegé.

II. *A normális integrál kiszámítása.*

Feladatunk az

$$I = \int_D \ldots \int \varphi(\underline{x})\, d\underline{x}$$

integrál kiszámítása. Az I becslésére használjuk a

$$\Theta_1 = \frac{1}{N} \prod_{j=1}^{n} (h_j + a) \sum_{i=1}^{N}{}' \varphi(\underline{\nu}^{(i)}) \tag{2.6}$$

valószínűségi változót, ahol $\underline{\nu}^{(i)}$-k valószínűségi vektorváltozók, melyek komponensei függetlenek és egyenletes eloszlásúak a $(-a, h_j)$, $j = 1, \ldots, n$ intervallumban, valamint maguk a $\underline{\nu}^{(i)}$-k is függetlenek egymástól. A (2.6) képlet lényegében az integrálandó tartomány "alapterületét" szorozza a sűrűségfüggvénynek az integrálandó tartományban felvett átlagértékével. Ezt a becslést integrálközépnek is nevezik.

Vizsgáljuk meg a (2.6) képleten alapuló eljárás hibáját. Θ_1 a

$$\xi_i = \prod_{j=1}^{n} (h_j + a) \cdot \varphi(\underline{\nu}^{(i)})$$

alakú valószínűségi változók számtani közepe:

$$\Theta_1 = \frac{1}{N} \cdot \sum_{i=1}^{N}{}' \xi_i \,.$$

Erre pedig ismert [1], hogy

$$M(\Theta_1) = I\,, \qquad D^2(\Theta_1) = \frac{D^2(\xi)}{N}\,,$$

$$P\{|\Theta_1 - M(\Theta_1)| \leqslant x_p D(\Theta_1)\} = p$$

valamilyen p valószínűségi szinttel (pl. $p = 0{,}997$-hez $x_p = 3$ tartozik). Igy valamilyen, p-hez közeli valószínűséggel igaz lesz a hibára:

$$(2.7) \qquad |\Theta_1 - I| \leqslant x_p \frac{D(\xi)}{\sqrt{N}} .$$

Tehát a módszer hibája a $\varphi(\underline{\nu}^{(i)})$ valószínűségi változók szórásától függ nagy mértékben és fordítottan arányos a mintapontok számának négyzetgyökével. Az eloszlást kiszámító szubrutin pontosságának növelése szempontjából éppen a $\varphi(\underline{\nu}^{(i)})$ szórásának csökkentése látszott tanácsosnak. A gépi program elkészítése során több, az irodalomban javasolt szóráscsökkentő eljárást kipróbáltam, de egyik sem bizonyult elég hatékonynak [2]; a mintapontok számának csökkenése ellenére a kiszámítás bonyolultsága miatt megnőtt a gépiidő azonos hiba mellett. Ezért lényegében az eredeti (2.6) képlet szerinti eljárást programoztam be, csak két további módosítást hajtottam végre az eljáráson.

Legyen K a következő n dimenziós kocka:

$$K = \{ \underline{x}: -\underline{a} \leqslant \underline{x} \leqslant \underline{a} \} .$$

Tekintettel arra, hogy $a = 4,5$ érték esetén elhanyagolható kis δ hibával igaz, hogy

$$\int_{R^n} \ldots \int \varphi(\underline{x})\, d\underline{x} = 1 = \int_{K} \ldots \int \varphi(\underline{x})\, d\underline{x} + \delta ,$$

ezért írhatjuk:

$$(2.8) \qquad \begin{aligned} I &= \int_{D^*} \ldots \int \varphi(\underline{x})\, d\underline{x} = 1 - \int_{R^n} \ldots \int \varphi(\underline{x}) d\underline{x} + \int_{D^*} \ldots \int \varphi(\underline{x}) d\underline{x} = \\ &= 1 - \left[\int_{R^n} \ldots \int \varphi(\underline{x})\, d\underline{x} - \int_{D^*} \ldots \int \varphi(\underline{x})\, d\underline{x} \right] = 1 - \int_{K-D^*} \ldots \int \varphi(\underline{x})\, d\underline{x} - \delta = \\ &= 1 - I_1 - \delta \approx 1 - I_1 . \end{aligned}$$

A $\approx$ jel azt jelenti, hogy az elkövetett hiba a kívánt pontossághoz és a kiszámításhoz felhasznált módszer egyéb hibáihoz képest nem jön számításba. I_1-et a következő valószínűségi változóval becsülhetjük:

$$(2.9) \qquad \Theta_2 = \frac{1}{N} \left| (2a)^n - \prod_{j=1}^{n} (h_j + a) \right| \cdot \sum_{i=1}^{N} \varphi(\underline{\nu}^{(i)}) ,$$

ahol $\underline{\nu}^{(i)}$ a $K - D^*$ tartományban egyenletes eloszlású véletlen pont. Vegyük még figyelembe, hogy a modell (1.2) felírásában szereplő p valószínűség értéke a gyakorlati alkalmazásokban $0.8 - 0.95$ szokott lenni, így a modell optimumát kiszámító algoritmus folyamán a kiszámítandó normális integrálok értéke ennél nagyobb, tehát a megfelelő I_1 érték $0.05 - 0.2$ között változik.

Azért jobb I helyett I_1-et, vagyis a (2.6) összeg helyett a (2.9)-et kiszámítani, mert a $\varphi(\underline{\nu}^{(i)})$ valószínűségi változónak a $K - D^*$ tartományban sokkal kisebb szórása van, mint a

D^* tartományban.

Ez azt jelenti, hogy azonos hiba mellett kevesebb véletlen pont felvételével kaphatjuk meg I értékét, ha I közvetlen kiszámítása helyett (2.9) segítségével megbecsüljük I_1-et és (2.8)-ból I_1 ismeretében meghatározzuk I értékét, vagyis a normális integrál kiszámításának hatékonysága nő. Szemléletesen a szóráscsökkenés azt tükrözi, hogy kihagytuk az átlagolandó összegből a sűrűségfüggvény által az origó közelében felvett csúcsot és a nagyobb értékeket.

Még egy módosítást hajtottam végre a normális integrál kiszámításában. A (2.9) átlag meghatározása a következő lépésekből tevődik össze a gépi programban.

1/ A program generál egy egyenletes eloszlású pszeudovéletlen pontot K-ban és megvizsgálja, hogy $K - D^*$-ba esik-e (ha nem, akkor új véletlen pontot számít ki, ellenkező esetben a 2/ lépés következik).

2/ Kiszámítja a sűrűségfüggvényben szereplő kitevőt, az $S = \sum_k' \sum_j' R_{kj} \nu_k^{(i)} \nu_j^{(i)}$ kvadratikus alakot.

3/ Elvégzi az $\exp\left\{-\frac{1}{2|R|} \cdot S\right\}$ érték meghatározását és hozzáadja Θ_2 eddig kiszámított részéhez.

Ha már N darab $\underline{\nu}^{(i)}$ vektor beleesett $K - D^*$-ba, akkor a Θ_2-ben gyűjtött összeget beszorozza a még hiányzó állandókkal.

Ez a három lépés körülbelül egyenlő hosszú idő alatt játszódik le. Előfordulhat és ez az eredeti szubrutin második módosítása, hogy bizonyos esetekben a harmadik lépést kihagyjuk, anélkül, hogy ezzel számottevő hibát okoznánk. Ugyanis a (2.9) összegben bizonyos i indexekre a $\varphi(\underline{\nu}^{(i)})$ olyan kicsi lehet, hogy még a

$$(2.10) \qquad \frac{1}{N} \cdot \left[(2a)^n - \prod_{j=1}^{n} (h_j + a) \right] \cdot \varphi(\underline{\nu}^{(i)}) = \frac{1}{N} \cdot T \cdot \varphi(\underline{\nu}^{(i)})$$

érték is kisebb mint pl. $\frac{I_1}{N \cdot 1000}$. (Lényeges, hogy a $\varphi(\underline{\nu}^{(i)})$ nagyságának megállapítása még a 3/ lépés, az e hatvány kiszámítása előtt megtörténhet az $S \geqslant -2|C| \cdot \ln\left(\frac{I_1 \cdot (2\pi)^{\frac{n}{2}} \cdot |R|}{T \cdot 1000}\right)$ feltétel vizsgálatával.) Tehát ha ezeket elhagyjuk a (2.9) összegből – nem végezzük el a 3/ lépést – legfeljebb $\frac{I_1}{1000}$ hibát követünk el.

Ez a módosítás különösen akkor jelent nagy időnyereséget, ha az R korrelációs mátrixban nagy együtthatók vannak (0,7 – 0,9), akkor ugyanis a sűrűségfüggvény értékei a tengelyekre koncentrálódnak, ott nagyobb értékeket vesz fel, míg a tengelyektől távolabb nagyon kicsi lesz, így sok olyan $\underline{\nu}^{(i)}$ pont lesz, amelyre a $\varphi(\underline{\nu}^{(i)})$ értékeket nem számítjuk ki kicsinysége miatt.

b/ Az (1.5) félegyenesen való haladás.

Tekintsük a

$$\underline{x}_k + \lambda(\underline{x}_k^* - \underline{x}_k), \qquad \lambda \geqslant 0 \tag{2.11}$$

félegyenest és határozzuk meg ennek a félegyenesnek a

$$\begin{aligned} &G(\underline{x}) \geqslant p\ , \\ &\underline{a}_i'\underline{x} \geqslant b_i\ , \qquad i = 1, \ldots M \end{aligned} \tag{2.12}$$

feltételek által határolt tartományba eső részét és ezen a részen keressük meg az $f(\underline{x})$ célfüggvény minimumhelyét. A gépi kipróbálásban $f(\underline{x})$ egy lineáris függvény volt, így a fenti feladat a (2.11) félegyenesnek a (2.12) feltételek által határolt tartomány határával való P_h metszéspontjának meghatározására redukálódik.

Két eset lehetséges: P_h egy $\underline{a}_i'\underline{x} = b_i$ feltétel által meghatározott hipersíkon fekszik, vagy P_h a $G(\underline{x}) = p$ feltétel által adott felületen található. Az első esetben könnyen meghatározható a P_h pont, ha tudjuk, hogy melyik hipersíkon van, mert a

$$P_h = \underline{x}_k + \lambda_h(\underline{x}_k^* - \underline{x}_k)$$

pontot helyettesítve a hipersík egyenletébe, a kapott

$$\underline{a}_i'(\underline{x}_k + \lambda_h(\underline{x}_k^* - \underline{x}_k)) = b_i$$

egyenletből λ_h meghatározható. A második eset sokkal nehezebb, t.i. nem lehet elemi úton a $G(\underline{x})$ függvény (a normális eloszlásfüggvény) inverzét kiszámítani és így λ_h-t közvetlenül megkapni, ezért ebben az esetben másképp kell eljárni. Előbb azonban vizsgáljuk meg, hogyan lehet eldönteni, hogy az első vagy a második esettel állunk szemben egy adott (2.11) félegyenes esetén.

Tekintsük az

$$\underline{a}_i'(\underline{x}_k + \lambda_i(\underline{x}_k^* - \underline{x}_k)) = b_i\ , \qquad i = 1, \ldots, M$$

egyenleteket a λ_i ismeretlenekkel, innen azt kapjuk, hogy

$$\lambda_i = \frac{b_i - \underline{a}_i'\underline{x}_k}{\underline{a}_i'(\underline{x}_k^* - \underline{x}_k)}\ , \qquad i = 1, \ldots, M\ .$$

Válasszuk ki ezek közül a legkisebb pozitívat, a

$$\lambda_{\min} = \min_{i:\lambda_i > 0} \lambda_i$$

értéket (a negatív λ_i értékek a félegyenesnek a másik irányban lévő hipersíkokkal való metszéspontjai; nyilván a legkisebb pozitív λ_i adja az $\underline{x}_k$-hoz legközelebbi hipersíkot, az elsőt,

amelyet a félegyenes metsz), és tekintsük a következő pontot:

$$P_m = \underline{x}_k + \lambda_{\min}(\underline{x}_k^* - \underline{x}_k) . \tag{2.13}$$

Ha $G(P_m) \geqslant p$, akkor nyilván az első esettel állunk szemben és ekkor $P_m = P_h$, egyébként a második esettel van dolgunk. Tekintettel arra, hogy $G(P_m)$ kiszámítása pontatlan, itt egyrészt az átlagosnál nagyobb pontosságú számítást kell megkövetelni, másrészt, ha tudjuk hogy $G(P_m)$ kiszámítása ϵ hibával történik, akkor csak a

$$G(P_m) \geqslant p + \epsilon$$

egyenlőtlenség teljesülése esetén döntünk úgy, hogy az első esettel állunk szemben.

Térjünk vissza arra, hogyan találjuk meg a P_h pontot, ha az a (2.11) félegyenesnek a $G(\underline{x}) = p$ egyenlet által megadott felülettel való metszéspontja. Az $\underline{x}_k$ pontból kiindulva a (2.11) félegyenesen haladva veszünk több, egymásutáni P_i pontot és vizsgáljuk, hogy mikor hagyja el a (2.12) feltételek által adott tartományt, helyesebben mikor lesz

$$G(P_i) < p .$$

$G(\underline{x})$ kiszámításának lassúsága miatt törekednünk kell arra, hogy a lehető legkevesebb P_i pontot válasszuk ki, mert mindegyiken ki kell számítani a $G(P_i)$ értéket. A legcélszerűbbnek a következő eljárás tűnt. Kiindulunk az $\underline{x}_k$ pontból, legyen például $\epsilon = 8$. Növeljük meg $\delta = \epsilon$-nal a $\lambda_0 = 0$ értéket egészen addig, amíg a keletkezett új pont:

$$P_j = \underline{x}_k + j \cdot \delta \cdot (\underline{x}_k^* - \underline{x}_k)$$

nem esik a $G(\underline{x}) \geqslant p$ térrészen kívül, vagyis legyen P_j az első olyan pont a P_i, $i = 1, \ldots$ $\ldots, n, \ldots$ pontok közül, amelyre

$$G(P_j) < p .$$

Ekkor vegyük az előző pontot, P_{j-1}-et kiindulópontnak, (ekkor $\lambda_1 = (j-1)\epsilon$) megfelezzük a lépésközt, vagyis $\delta = \epsilon/2$-t veszünk és tovább lépünk: λ_1-hez $\delta = \epsilon/2$-t adunk, amíg a keletkezett

$$P_i = \underline{x}_k + (\lambda_1 + i \cdot \epsilon/2)(\underline{x}_k^* - \underline{x}_k)$$

pont nem hagyja el a (2.12) tartományt, vagyis amíg

$$G(P_i) < p$$

nem lesz. Ekkor $\lambda_2 = \lambda_1 + (i-1) \cdot \epsilon/2$, ezt az újból felezett lépésközzel $\delta = \epsilon/4$-gyel növeljük, kiindulópontként P_{i-1}-et használva, stb. Természetesen, ha $i = 0$ adódna, akkor P_{i-1} helyett a P_{j-1} pontot vesszük kiindulópontnak. Ezt az eljárást ismételjük egészen $\delta = \epsilon/2^6$-ig. (Egy adott tartomány és félegyenes esetén ez a legkisebb lépésszámú megközelítés, abban az

értelemben, hogy ha nem feleznénk a lépésközt, hanem harmadolnánk, akkor több lépést kellene tennünk.)

Ha egy adott k iterációszám esetén $\lambda_6 < 0{,}5$ adódik, akkor $\epsilon = 0{,}2$-t veszünk és az egész eljárást megismételjük. Erre azért van szükség, hogy finomabb lépéseket téve közelebb kerüljünk a határhoz, pontosabb eredményt kapjunk P_h-ra. Ha $\underline{x}_k^* - \underline{x}_k$ kicsi, akkor egy nagyobb ϵ is megfelel; a konvergencia miatt $\underline{x}_k^* - \underline{x}_k$ állandóan csökken k növekedésével, így a fentebbi lépegető eljárásban egyre kisebb lépéseket teszünk, így önmagától növekszik a határ megközelítésének pontossága. A gyakorlatban viszont kiderült, hogy az $\underline{x}_k^* - \underline{x}_k$ egy bizonyos határon túl nem csökken k növekedtével, így célszerűnek mutatkozott egy kisebb ϵ bevezetése.

A fentebb leírt lépegető eljárás egész jól működne, ha $G(\underline{x})$ értékét pontosan ki lehetne számítani. Egy reális gépi idő alatt (a CDC 3300-as gépen 0,8 sec alatt 1000 mintapontot véve) a $G(\underline{x})$ értékét a kétdimenziós esetben 5 %-os hibával lehet csak kiszámítani. Igy előfordulhat, hogy a félegyenesen lépegetve már elhagytuk a $G(\underline{x}) \geqslant p$ egyenlőtlenséggel megadott térrészt, csak a hiba miatt még mindig a $G(P_i) > p$ teljesül, illetőleg megeshet, – ami kisebb baj – hogy jóval a $G(P_i) = p$ felület előtt megállunk, de ilyenkor a kisebb δ-k még csökkenthetik ezt a távolságot. Arra az esetre, ha már kiléptünk a megengedett megoldások tartományából, a következő két ellenintézkedést programoztam be.

Ha azt tapasztaljuk, hogy egy adott $\delta = \epsilon/2^i$ esetén P_j az első pont, amely már a határon kívül esik, vagyis $G(P_j) < p$, akkor nem a P_{j-1}-et, az előző pontot vesszük újabb kiindulópontnak, hanem a még eggyel előbbit, vagyis P_{j-2}-t, ilyenkor $\lambda_i = \lambda_{i-1} + (j-2)\delta$ lesz. Ennek az az előnye, hogy ha már a P_{j-1} pont is kívülesett, csak éppen a szubrutin hibás eredményt adott, akkor nagyon valószínű hogy a P_{j-2} pont esetében nem, vagy az ellenkező irányban hibázott a szubrutin, vagyis a P_{j-2} még bent van a tartományban.

Nyilván a fentebbi eljárás beprogramozása még nem nyújt biztosítékot arra nézve, hogy nem kerülünk ki a (2.12) feltételek által határolt tartományból.

Ha kikerültünk, ezt két jelből lehet észrevenni: a kapott $\underline{x}_{k+1} = P_h$ pontot véve új megengedett megoldásnak, amire valójában $G(\underline{x}_{k+1}) < p$, az ebből képzett új lineáris feladat optimális megoldásában $y > 0$, valamint azt tapasztaljuk, hogy az ebből képzett (2.11) félegyenesen haladva a minimum értéke nem csökken, hanem növekszik.

Ekkor csináljuk a következőt: tekintjük az $\underline{x}_k$ és az $\underline{x}_{k+1}$ pontokat összekötő szakaszt és ezen a szakaszon visszalépünk a következő szabály szerint; ha a szakasz hossza kisebb 0,2-nél, akkor $q_{\min} = 0{,}2$-t veszünk, ellenkező esetben $q_{\min} = 0{,}05$-öt véve új megengedett $\underline{x}_{k+2}$ megoldást veszünk a rossznak bizonyult $\underline{x}_{k+1}$ helyett:

$$\underline{x}_{k+2} = q_{\min}\underline{x}_k + (1 - q_{\min})\underline{x}_{k+1} ,$$

vagyis visszalépünk az $\underline{x}_k - \underline{x}_{k+1}$ szakaszon $\underline{x}_{k+1}$-ből $\underline{x}_k$ felé a szakasz hosszának ötödrészével vagy huszadrészével.

Ezzel az $\underline{x}_{k+2}$ új megengedett megoldással újra készítünk egy lineáris problémát. Ha még mindig $y > 0$, vagy a minimum növekedne az

$$\underline{x}_{k+2} + \lambda(\underline{x}^*_{k+2} - \underline{x}_{k+2}), \qquad \lambda > 0$$

félegyenes mentén, akkor $\underline{x}_{k+2}$ helyett az

$$\underline{x}_{k+3} = 2q_{\min}\underline{x}_k + (1 - 2q_{\min})\underline{x}_{k+1}$$

értéket tekintsük új megengedett megoldásnak, stb. Ez a két vészkijárat elegendőnek mutatkozott a gyakorlatban arra, hogy a program normálisan lefusson.

A fenti módosított lépegető algoritmusnak, nevezetesen a két ponttal való visszalépésnek egy nagy előnye, hogy ilyen módon automatikusan pontosabban számol a program, mert a $G(\underline{x}) = p$ felület közelében sokkal többször számítja ki az eloszlásfüggvény értékét, így végső soron a mintaszám növekszik.

Nyilvánvaló, hogy az egész eljárás hatékonysága attól függ, hogy a $G(P_i)$ értékeket milyen pontosan számítom ki. A gépidő csökkentése szempontjából pedig ajánlatos lenne minél kisebb pontossággal (minél kevesebb véletlen pontot felvéve) meghatározni a normális eloszlásfüggvény értékét a P_i pontokban. Mivel a lépegető eljárás folyamán csak a $G(P_i) > p$ feltétel teljesülésének eldöntéséről van szó, a valószínűséget "szekvenciálisan" fogom meghatározni. Ez azt jelenti, hogy először kiszámítom a $p_1 = G(P_i)$ értéket 10 %-os hibával, amihez kevés véletlen pont és gépiidő szükséges. Ha $|p_1 - p| > 0{,}15$ (a biztonság kedvéért 0,1 helyett 0,15-öt írunk, mivel a normális eloszlás konkrét értékét kiszámító rutinról csak azt mondhatom, hogy elég nagy valószínűséggel lesz legfeljebb 10 %-os a hiba), akkor döntök a $p_1 = G(P_i) > p$ egyenlőtlenség teljesülése felől. Ha p_1 és p eltérése kisebb mint 0,15, akkor kiszámítom a $p_2 = G(P_i)$ értéket 5 %-os hibával és vizsgálom a $|p_2 - p| > 0{,}08$ egyenlőtlenséget; a teljesülés esetén döntök $p_2 > p$ felől. Ellenkező esetben megint fokozom a pontosságot, 2,5 %-os hibával számítom ki a normális eloszlásfüggvény értékét és ekkor mindenképpen döntök, hogy a $G(P_i) > p$ feltétel teljesül-e.

Figyelembe kell még venni, hogy amíg messze vagyunk az eredeti (1.3) probléma optimális megoldásától, nem érdemes nagyon pontosan meghatározni a (2.11) félegyenesnek a (2.12) feltételek által határolt tartomány határával való P_h metszéspontját. Ezért a programot három fázisban futtatjuk le. Az első fázisban a lépegető eljárás folyamán csak 10 %-os hibával határozzuk meg a $G(P_i)$ értékeket. A második fázisban 10 %-os vagy 5 %-os pontossággal, attól függően, hogy a szekvenciális valószínűségvizsgálat szükségessé teszi-e. A harmadik fázisban pedig 10 %-os, 5 %-os, 2,5 %-os pontossággal határozzuk meg a szükségleteknek megfelelően a $G(P_i)$ függvényértékeket. Az egyes fázisok befejezésének feltételeit a következő alpontban fogjuk megvizsgálni.

c/ Az iterálás befejezésének feltétele.

Azt kell megvizsgálni, hogy mikor jutunk már elég közel az optimális megoldáshoz, milyen kritérium alapján állítsuk meg az iterálást. Az első kézenfekvő kritérium, ha két egymásutáni megengedett megoldáson a célfüggvény értéke kicsivel, pl. a célfüggvény értékének 1 %-ával változna csak, akkor megállhatunk. De ez önmagában nem elég, mert az általunk betáplált első megoldás és a gép által kiszámított első megengedett megoldás ennek a feltételnek mindig eleget tesz.

Ezért célszerűnek mutatkozott a következő kritériumot másodikként bevenni, nevezetesen azt, hogy az utolsó két megengedett megoldás egyes koordinátáinak eltérése kisebb legyen mint a megengedett megoldásvektor nagyságának pl. 1 %-a. Ezzel viszont az volt a probléma, hogy ha a félegyenes a valóságban egy $\underline{a}'_i\underline{x} = b_i$ hipersíkot metszett, az eloszlást kiszámító szubrutin hibázott és azt mutatta, mintha ez a metszéspont már a megengedett megoldások halmazán kívül lenne, akkor a lépegető eljárásra kerül sor és könnyen elképzelhető, hogy valamivel a hipersík előtt meg fog állni. A következő iterációs lépésben majdnem ugyanazt az irányt fogjuk a (2.11) félegyenesnek kapni és így a következő megengedett megoldás valamivel még közelebb kerül a hipersíkhoz. A másodikként említett kritérium alapján ezt a két egymásutáni megoldást már optimálisnak venné a program.

Nagyon könnyen előfordulhat az is, főleg az optimális megoldáshoz közel, hogy $\lambda_6 = 0$ adódik, vagyis már nem tud továbblépni az eljárás, a fenti kritérium ekkor is megállítaná a programot. Ezen esetek kiküszöbölése végett csak három egymásutáni megengedett megoldás fentiekben megfogalmazott "közelsége" esetén fogjuk megállítani az iteráló eljárást.

A b/ pontban említett fázisok végét úgy állapítjuk meg, hogy a fenti kritériumokat alkalmazzuk a következő módon. Az első fázis akkor ér véget, ha a célfüggvény értéke az utolsó két megengedett megoldáson legfeljebb 2 %-kal változik és a három legutolsó megengedett megoldás fentiekben leírt eltérése szintén csak 2 %-os. A második fázisban már 1 %-ot követelünk meg, a harmadikban pedig 0,5 %-ot.

3. A MODELL KIÉRTÉKELÉSE, SZÁMITÁSI EREDMÉNYEK

A modell kiértékelése több adatcsoport segítségével történt. Minden adatcsoportban azonban a $g_i(\underline{x})$ függvények és az $f(\underline{x})$ célfüggvény az egyszerűség kedvéért lineáris volt, két sztohasztikus és két lineáris determinisztikus feltételt vettem, az $\underline{x}$ pedig két dimenziós vektor volt.

A próbák megmutatták, hogy ha a $G(\underline{x}) = p$ által generált felület a lineáris feltételek által generált térrészen kívül esik (vagyis a $G(\underline{x}) \geqslant p$ térrész tartalmazza az $\underline{a}'_i\underline{x} \geqslant b_i$ feltételek által adott poliédert; legalábbis a miáltalunk vizsgált részen), akkor a módszer szimplex algoritmussá zsugorodik, a poliéder csúcspontjain mozogva éri el az optimumot. Ellenkező esetben

is, amíg a normális eloszlásfüggvény nívófelületébe nem ütközik, csúcspontokon haladva működik az algoritmus.

A módszer jobb bemutatása érdekében egy olyan példát vettem illusztrálásként, amelyen a $G(\underline{x}) \geqslant p$ feltétel által meghatározott térrész benne van a lineáris feltételek által adott poliéderben, vagyis a valószínűségi feltétel "megtartja" az egymásutáni megengedett megoldásokat. Az algoritmus gyorsasága, helyesebben az iterálások száma függ attól, hogy milyen "meredek" a $G(\underline{x}) = p$ felület; gyakorlatilag ez azzal mérhető le, hogy a $g_i(\underline{x}) \geqslant \beta_i$ feltételek (itt β_i valós) által adott hipersíkok milyen szögben metszik egymást.

A bemutatott eredmények a következő modellre vonatkoznak:

$$p\left\{\begin{matrix} 2x_1 + x_2 - 6 \geqslant \beta_1 \\ x_1 + 8x_2 - 8 \geqslant \beta_2 \end{matrix}\right\} \geqslant p\,,$$

$$x_1 + 4x_2 \geqslant 4\,,$$

$$3x_1 + x_2 \geqslant 3\,,$$

$$x_1 \geqslant 0_1\,,\ x_2 \geqslant 0\,,$$

$$\min(x_1 + x_2)\,.$$

Itt β_1 és β_2 együttes eloszlása normális, nulla a várható értékük, 1 a szórásuk és r a korrelációs együtthatójuk. Ha a kiindulásul vett determinisztikus problémát vizsgáljuk, vagyis a

$$3x_1 + x_2 \geqslant 6\,,$$

$$x_1 + 8x_2 \geqslant 8,$$

$$x_1 + 4x_2 \geqslant 4\,,$$

$$3x_1 + x_2 \geqslant 3\,,$$

$$x_1 \geqslant 0\,,\ x_2 \geqslant 0\,,$$

$$\min(x_1 + x_2)$$

feladatot, az optimális megoldásra és a minimalizálandó célfüggvény értékére azt kapjuk, hogy

$$\underline{x}_{\text{opt}} = \left(\frac{40}{23}, \frac{18}{23}\right) \sim (1{,}739,\ 0{,}782)\,, \qquad f(\underline{x}_{\text{opt}}) \sim 2{,}521\,,$$

ami lényegesen különbözik a sztohasztizált modell eredményeitől, akármilyen korrelációs együtthatót véve.

A következő értékeket kaptuk a próbafuttatások alkalmával az $\underline{x}_{\text{opt}}$ optimális vektorra és az $f(\underline{x}_{\text{opt}})$ célfüggvény értékre $p = 0{,}8$ valószínűség mellett, változó r korrelációs együtthatóra:

r	$\underline{x}_{opt}$		$f(\underline{x}_{opt})$
– 0,9	1,878	1,015	2,893
– 0,2	1,942	1,001	2,943
+ 0,2	1,984	0,964	2,948
+ 0,5	2,006	0,944	2,950
+ 0,9	2,065	0,888	2,953

Látható, hogy a korrelációs együttható növelésével nő az optimum értéke, ami összhangban van azzal a valószínűségszámítási eredménnyel, hogy a korrelációs együttható növelésével csökken egy adott pontban az eloszlásfüggvény értéke (hasonló tétel többdimenzióban is igaz.)

A $p = 0,95$ esetre is kipróbáltam a programot, ez az eset azért fontos, mert a modell egyik alkalmazási területe a megbízhatóságelmélet lenne. A kapott eredmények:

r	$\underline{x}_{opt}$		$f(\underline{x}_{opt})$
+ 0,2	2,161	0,961	3,1227
+ 0,9	2,279	0,931	3,2108

A fentebbi két táblázatot a program ALGOL nyelvű változatának lefuttatásával kaptam, ebben a normális eloszlás konkrét értékeit a (2.6) képlet segítségével számítottam. Egy-egy adatcsoport lefuttatásához átlag 20 percre volt szükség a CDC 3300-as gépen. Különösen sokáig tartott a $p = 0,95$, valamint a nagy korrelációjú esetek lefuttatása (30-35 perc), ez annak tulajdonítható, hogy ezekben az esetekben a normális eloszlást számító rutin a legkevésbé hatékony.

A program FORTRAN nyelvbe való átírása, valamint a normális eloszlásfüggvények a (2.9) képlet alapján történő kiszámítása kb. 5 percre csökkentette a fentebbi konkrét példa futási idejét. Az eloszlásfüggvény konkrét értékét kiszámító módosított szubrutin különösen azokban az esetekben számol gyorsan és viszonylag pontosabban, amelyekben az előző (2.6) alapján működő rutin nehézkesen és sokáig számolt (nagy valószínűség, nagy korreláció) [2]. Ez a (2.9) képlet alapján történő számításnak és a 2b/-ben leírt két másik módosításnak köszönhető.

Néhány adatot közlünk még a normális eloszlásfüggvénynek adott pontban felvett értékét kiszámító szubrutin működési gyorsaságáról. Az eredeti, ALGOL nyelven a (2.6) képlet alapján működő szubrutin 3 sec alatt számított ki 1000 mintapont segítségével egy eloszlásértéket a kétdimenziós esetben a CDC 3300 gépen, ennek hibája a $p = 0,8$ érték közelében 6 %- 14 % volt a korreláció nagyságától függően. Teljesen ugyanezt a szubrutint megírva FORTRAN nyelven a futási idő 1 sec-ra csökkent. A legújabb szubrutin, amely a (2.9) képlet alapján dolgozik és minden 2b/-ben megemlített módosítást tartalmaz, 0,1 sec alatt számít ki egy kétdimenziós eloszlásfüggvényértéket 5 %-os hibával. Ez a hiba független a korrelációtól, de a valószínűség értékétől nem; nagyobb valószínűségre hatékonyabban számol. Ezt a szubrutint már magasabb dimenzióban is kipróbáltam: az ötdimenziós esetben $p = 0,9$ valószínűség esetén 5 %-os hibával 0,6 sec alatt számított ki egy értéket.

Megjegyezzük még, hogy a program további gyorsítására egy speciális szimplex írása lenne. Ugyanis az egymásutáni iterációs lépésekben megoldandó lineáris programozási feladatok egymástól csak két sorban különböznek, ami valószínűsíti, hogy optimális megoldásaik sem különböznek nagyon és ezt fel is lehet használni.

Irodalom

[1] N.P. Buszlenko – D.I. Golenko – Ju.A. Srejgyer – I.M. Szobol – V.G. Szragovics, Monte Carlo módszerek. Műszaki Könyvkiadó 1965.

[2] Deák István, Egy sztohasztikus programozási modell számítógépes kiértékelése. (Egyetemi doktori disszertáció ELTE TTK 1971.)

[3] S.S. Gupta, Probability integrals of Multivariate Normal and t distribution, Ann. Math. Stat. 34 (1963) N. 3 pp. 792-829.

[4] Prékopa A., Sztohasztikus rendszerek optimalizálási problémáiról. (Doktori értekezés, TMB. 1970)

[5] G. Zoutendijk, Methods of feasible directions. Elsevir publishing company, Amsterdam–London–New York–Princeton (1960).

Beérkezett: 1972 március 15.

Summary

Computer evaluation of a stochastic programming model

The paper discusses a stochastic programming model, developed by A. Prékopa and tested on a computer. Chief results of the article are the elimination of the difficulties of computer application and the description of an efficient algorithm for the computation of the concrete values of the multidimensional normal distribution function.

Р е з ю м е

Оценка одной проблемы стохастического программирования

В статье даётся оценка одной проблемы стохастического программирования, разработанной А. Прекопа, с помощью ЦВМ. Преодоление трудностей при применении ЦВМ и описание эффективного алгоритма для получения конкретных значений многомерной функции нормального распределения являются главными результатами работы.