Kovács László Béla

A diszkrét programozás módszerei és alkalmazása gazdasági problémák megoldására

Gazdasági problémák megoldásában nagy szerepet játszik a matematikai programozás. Ennek feladata, hogy egy adott többváltozós függvény minimumát vagy maximumát és annak helyét meghatározza egy megadott tartományon. Ha a maximalizálandó függvény — az ún. célfüggvény — lineáris, továbbá a tartomány lineáris egyenlőtlenségek segítségével van megadva (vagy ilyen alakra hozható), akkor lineáris programozásról beszélünk. A lineáris programozás matematikai elméletét, valamint alkalmazásokat találhat az olvasó az alábbi könyvekben: Prékopa A. [36], Hadley [20], Dantzig [13], Charnes és Cooper [11]. Az említett művekben bőségesen vannak további irodalmi utalások. Gyakran a lineáris közelítés pontossága már nem megfelelő. Ha a célfüggvény vagy a feltételek nem lineárisak, akkor nemlineáris programozásról beszélünk. Exakt módszerek jelenleg többnyire csak arra az esetre vannak, amikor a maximalizálandó célfüggvény konkáv, esetleg kvázikonkáv (vagy a minimalizálandó függvény konvex, ill. kvázi-konvex) és a határoló feltételek is konvex tartományt szolgáltatnak. Ha a feladat nem ilyen, akkor az okozza a nehézséget, hogy sok lokális (helyi) maximum lehetséges, amelyek nem globális maximumok. A nemlineáris programozás jó áttekintése néhány módszer alapos tárgyalásával megtalálható Hadley [21] könyvében, amelyben dinamikus programozás is található. Itt a „dinamikus" kifejezés nem feltétlenül ugyanazt jelenti, mint a közgazdaságtanban. Általában arra utal, hogy több egyszerre megteendő döntés helyett a döntések egymásutánjára vezeti vissza a problémát. Az újabb döntéseket a korábbiak következtében kialakult helyzetben optimálisan teszi meg. Lényegében tehát a dinamikus programozás egy bonyolult problémát egyszerűbb feladatok sorozatára vezet vissza. A dinamikus programozás alapos matematikai tárgyalása, valamint nagyszámú alkalmazás található Bellman[7], Bellman és Dreyfus [8] könyvében. Bevezetésként Neuhauser [34] műve ajánlható.

Ha a matematikai programozási feladatban valószínűségi változók is szerepelnek (igény, technikai koefficiensek, árak stb.) akkor sztochasztikus programozásról beszélünk. (Lásd: Dantzig [13]. 25. fejezet, Hadley [21]. 5. fejezet, Prékopa [37].)

Mind a lineáris mind a nemlineáris feladatokban, ha külön nem kötjük ki, általában folytonos változókra gondolunk. Sok esetben azonban egyes változók nem tekinthetők folytonosnak. Például nagyértékű gépek esetén nem mindegy, hogy az eredményül kapott 3,3 gépet 3 vagy 4-nek tekintjük. Különösen kritikus a helyzet az ún. beruházási problémákban, ahol a változó értéke 1 vagy 0 aszerint, hogy egy beruházást végrehajtunk-e vagy sem.

Az első részben a diszkrét programozási feladatok általános alakját, a má-

sodikban néhány gazdasági alkalmazást tárgyalunk, míg a harmadik rész a diszkrét programozási módszerek áttekintését tartalmazza irodalmi utalásokkal.

1) A diszkrét programozási feladatok általános alakja

Diszkrét programozásról beszélünk olyan matematikai programozási feladat esetén, amelyben néhány vagy valamennyi változó csak véges vagy megszámlálhatóan végtelen sok különböző értéket* vehet, más szóval értelmezési tartománya nem folytonos.

Diszkrét programozás helyett használatos még az egészértékű és az integer programozási elnevezés is.

A két utóbbi elnevezés magába foglalja azt a megkötést is, hogy a nem folytonos változók csak egész értékeket vehetnek fel, azonban ez nem jelent megszorítást. Ha ugyanis az x változó az

$$a_1, a_2, \ldots$$

véges vagy megszámlálhatóan sok (nem feltétlenül egész) értéket veheti fel, akkor az

$$x = a_1 x_1 + a_2 x_2 + \ldots$$

$$x_1 + x_2 + \ldots = 1$$

$$x_j = 0 \text{ vagy } 1$$

felírásban már csak egészértékű változók szerepelnek.

Továbbiakban ezért a diszkrét változók csak egész értékeket vesznek fel. Ha egy feladatban csak egészértékű változók szerepelnek, akkor ezt *tiszta* egészértékűnek, egészértékű és folytonos változók fellépése esetén *vegyes* vagy *kevert* egészértékű feladatnak nevezzük.

Továbbá egy tetszőleges egészértékű feladatot is át lehet alakítani $0-1$ egészértékűvé, ha valamennyi változó korlátos. Legyen például

$$x = 0, 1, 2, \ldots, p$$

Természetesen adódik az

$$x = y_1 + y_2 + \ldots y_p$$

$$y_j = 0 \text{ vagy } 1 \; (j = 1, \ldots, p)$$

előállítás, ennél azonban sokkal jobb is van.

Határozzuk meg a k számot úgy, hogy

$$2^{k-1} \leq p < 2^k$$

* Például tetszőleges egész vagy nemnegatív egész értékeket vehet fel.

ekkor legyen

$$x = 2^{k-1}x_1 + 2^{k-2}x_2 + \ldots + 2x_{k-1} + x_k$$

ahol az

$$x_j = 0 \text{ vagy } 1 \quad (j = 1, 2, \ldots, k)$$

Ennek az előállításnak az a jó tulajdonsága is megvan, hogy az x változó és az $(x_1, x_2, \ldots, x_k)$ vektor változó között egy-egy értelmű megfeleltetés létesíthető és mindössze

$$\left[\underbrace{\log p}^{2}\right]$$

számú változó szükséges hozzá.

A leggyakrabban előforduló típus a lineáris egészértékű feladat, mely a következőképpen írható fel.

$$\min (c_1x_1 + c_2x_2 + \ldots + c_nx_n)$$

$$a_{11}x_1 + a_{12}x_2 + \ldots a_{1n}x_n \geqq b_1$$

$$a_{21}x_1 + a_{22}x_2 + \ldots a_{2n}x_n \geqq b_2$$

$$\vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \geqq b_m$$

$$x_j = 0, 1, 2, \ldots, r_j \qquad (j = 1, 2, \ldots, k)$$

$$x_j \geqq 0 \qquad (j = k+1, \ldots, n)$$

ahol a_{ij}, b_i és c_j adott tetszőleges valós számok, az r_j, k számok adott pozitív egészek. A $k = n$ esetben tiszta, $0 < k < n$ esetén kevert egészértékű feladatról beszélünk.

Ha a célfüggvényt maximalizálni kell, annak (-1)-szeresét minimalizálhatjuk. A $\leqq$ egyenlőtlenségeket (-1)-gyel való beszorzással $\geqq$ típusúvá tehetjük, míg az egyenlőségeket felbonthatjuk egy $\leqq$ és egy $\geqq$ típusú egyenlőtlenségre.

Ha egyes egészértékű változókra nincs explicit felső korlát, de az egyenlőtlenségek által határolt tartomány korlátos, akkor ebből célszerű megállapítani a szóban forgó felső korlátot, ez megkönnyíti a megoldást.

Nem okoz nehézséget az a változó sem, amelyik negatív értékeket is felvehet, de sokszor egyöntetűség, illetve könnyebb kezelhetőség kedvéért ezeket felbonthatjuk pozitív és negatív részükre. Ha eredetileg x_j előjelkötetlen változó volt, akkor

$$x_j = x_j^+ - x_j^-$$

$$x_j^+ \geqq 0, \qquad x_j^- \geqq 0,$$

ha pedig

$$x_j = \ldots, -2, -1, 0, +1, +2, \ldots,$$

akkor

$$x_j = x_j^+ - x_j^-$$
$$x_j^+ = 0, 1, 2, \ldots$$
$$x_j^- = 0, 1, 2, \ldots$$

Bevezetve az

$$\mathbf{A} = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ a_{m1} & \ldots & a_{mn} \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b^1 \\ \cdot \\ \cdot \\ \cdot \\ b_m \end{pmatrix}, \quad \mathbf{c} = \begin{pmatrix} c_1 \\ \cdot \\ \cdot \\ \cdot \\ c_n \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix}$$

jelöléseket, az (1) feladat a következő egyszerű formában írható fel:

$$\text{(2)} \qquad \min \mathbf{c}^T\mathbf{x}$$
$$\mathbf{A}\mathbf{x} \geq \mathbf{b}$$
$$x_j = 0, 1, \ldots, r_j \qquad (j = 1, 2, \ldots, k)$$
$$x_j \geq 0 \qquad (j = k+1, \ldots, n),$$

ahol a felső indexben szereplő T transzponáltat jelent. (Egy mátrix transzponáltját főátlójára való tükrözéssel, azaz a sorok és oszlopok felcserélésével kapjuk. Így egy oszlopvektor transzponáltja sorvektor, a sorvektor transzponáltja oszlopvektor. A továbbiakban a vektorok általában oszlopvektort jelentenek, hacsak külön másképpen nem kötjük ki.)

Hasonlóan írható fel a megfelelő nemlineáris probléma:

$$\text{(3)} \qquad \min \quad g_0(\mathbf{x})$$
$$g_i(\mathbf{x}) \geq 0 \qquad (i = 1, 2, \ldots, m)$$
$$x_j = 0, 1, \ldots . r_j \qquad (j = 1, 2, \ldots, k)$$
$$x_j \geq 0 \qquad (j = k+1, \ldots, n),$$

ahol $g_i(\mathbf{x})$ $(i = 0, 1, \ldots, m)$ tetszőleges (egyértékű) függvények. Más kérdés természetesen, hogy ilyen általánosságban a feladat nem oldható meg, még akkor sem, ha csak egészértékű változók szerepelnek és a feladat méretei túl nagyok a teljes leszámlálásra. A későbbiekben adunk a tiszta egészértékű változatra a $g_i(\mathbf{x})$ függvényekre vonatkozó, nem túl erős feltételek mellett algoritmust, mely elég nagyméretű feladatokra alkalmazható.

Végül meg kell említeni, hogy az egészértékű változókra igen gyakran bizonyos típusú logikai feltételek is fennállnak. Ezek igen jól beleilleszthetők a leszámlálási típusú és a korlátozás és szétválasztáson alapuló algoritmusokba, sokszor még meg is gyorsítják azt.

Ilyen feltétel például, hogy nem lehet három egymás után következő egészértékű változó zérustól különböző, vagy az egészértékű változók egy csoport-

jának nem zérus értékéből következik, hogy másoknak zérus értéket kell felvenni. Az ilyen típusú fetételek a legtöbb esetben megfogalmazhatók algebrai alakban is, azonban rendszerint csak igen nagyszámú feltétellel, ami két hátránnyal is jár. Az egyik az, hogy a sok feltétel igen sok helyet foglal el (gépi kapacitásban), a másik pedig, hogy a feltételek ellenőrzése nagyon sokáig tart. Míg, ha meg tudunk maradni a logikai kezelésmód mellett, akkor ezt úgy foghatjuk fel, mintha a fenti nagyszámú feltételnek mindig csak egy kis részét vizsgálnánk, mert a többiről egyszerűen ki tudjuk mutatni, hogy jelenleg nem jelentenek korlátozást.

2) A diszkrét programozás néhány gazdasági alkalmazása

A jelen szakaszban a teljesség igénye nélkül felsorolunk néhány alkalmazást. A modelleket lehetőleg egyszerű formában tárgyaljuk a könnyebb megértés kedvéért, ez azonban nem jelenti azt, hogy komplex problémák megoldására nem alkalmasak.

2.1 *Egy feltételes terhelési feladat*

Az irodalomban ez „hátizsák probléma" néven ismeretes, mert az első megfogalmazása egy gyalogtúrázó hátizsákjának optimális kitöltése volt egy megadott választékból. Ez azt jelenti, hogy egy adott súlykorlát alatti tárgykombinációk közül a maximális használati értékűt keressük.[1]

Ez a probléma nemcsak egyszerűsége és önmagában való használhatósága miatt érdekes, hanem azért is, mert a többfeltételes probléma megoldásában is jó segédeszköz.

A következő jelöléseket használjuk:

n a tárgyak száma (ha valamelyikből több van, azt többszörösen használjuk)
a_j a j-edik tárgy súlya,
c_j a j-edik tárgy értéke (használati érték),
K a megengedett maximális terhelés,
$x_j = \begin{cases} 1 & \text{ha } j\text{-edik tárgyat választjuk,} \\ 0 & \text{ha nem.} \end{cases}$

A feladat a következőképpen fogalmazható meg:

$$(\max)\, z = \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=1}^{n} a_j x_j \leq K$$

$$x_j = 0 \text{ vagy } 1 \quad (j = 1, 2, \ldots, n).$$

[1] Feltételezzük, hogy a használati értékek összeadódnak. Csak egyszerűség kedvéért választottuk a feladatnak ezt a — gazdasági szempontból kifogásolható — értelmezését. Megfogalmazhattuk volna például úgy is, mint a 2.2 modellnek azt a speciális esetét, amikor csak egyetlen feltétel van.

Az általános alkalmazhatóság szempontjából megvizsgáljuk az együtthatók különböző előjelkombinációit. Feltételezhetjük, hogy $c_j > 0$ és $a_j > 0$ $(j = 1, \ldots, n)$. Ugyanis, ha $c_j < 0$ és $a_j < 0$, akkor az $x'_j = 1 - x_j$ változó bevezetésével az együtthatók pozitívvá válnak. A $c_j > 0$, $a_j \leq 0$ esetén $x_j = 1$ és $c_j \leq 0$, $a_j > 0$ esetén pedig $x_j = 0$, ugyanis az első esetben pozitív értéket kapunk újabb kapacitás felhasználása nélkül (sőt $a_j < 0$ esetén még növeljük is azt), a másodikban pedig bizonyos kapacitás felhasználásával biztosan nem növeljük az összes értéket, sőt esetleg csökkentjük. Végül a $c_j = a_j = 0$ eset teljesen érdektelen, mert az x_j változó ekkor nem szerepel a feladatban.

A feladat optimális célfüggvény-értékére jó becslést kaphatunk a következő módon. Állítsuk a változókat olyan sorrendbe, hogy

$$\frac{c_1}{a_1} \geq \frac{c_2}{a_2} \geq \ldots \geq \frac{c_n}{a_n}$$

(Egyszerűség kedvéért feltételezzük, hogy eleve ilyen sorrendben indexeztük a változókat.)

Ekkor, ha

$$\sum_{j=1}^{r} a_j \leq K < \sum_{j=1}^{r+1} a_j,$$

akkor

$$\sum_{j=1}^{r} c_j \leq \max z \leq \sum_{j=1}^{r} c_j + \frac{c_{r+1}}{a_{r+1}} \cdot \left(K - \sum_{j=1}^{r} a_j \right).$$

Az $x_1 = x_2 = \ldots = x_r = 1$, $x_{r+1} = x_n = 0$ a feladat megengedett megoldása és rendszerint elég jó célfüggvény-értéket szolgáltat. Arra azonban semmi biztosíték nincs, hogy ez optimális megoldás volna. Például a

$$(\max)\ z = 5x_1 + 3x_2 + 9x_3$$

$$2x_1 + 3x_2 + 10x_3 \leq 11$$

$$x_1, x_2, x_3 = 0 \text{ vagy } 1$$

feladat már rendezve van, mert

$$\frac{5}{2} > \frac{2}{3} > \frac{9}{10},$$

mégis az $x_1 = x_2 = 1$, $x_3 = 0$ megoldás célfüggvényértéke $z = 8$, míg az $x_3 = 1$, $x_1 = x_2 = 0$ esetén $z = 9$.

Az említett feladatnál az összes megoldások száma $2^3 = 8$, így könnyű valamennyit ellenőrizni. Nagyobb n esetén azonban ez nem lehetséges, mivel a 2^n igen gyorsan növekszik n növelése esetén.

2.2 *Több-feltételes terhelési feladat*

A nehézségek jól látszanak már két feltétel esetén is, így ezzel az esettel foglalkozunk. Tegyük fel, hogy egy hajót kell terhelnünk értékes rakománnyal. Az érték most lehet használati érték (expedíció), valódi érték (mentési munkálatok) vagy szállítási költség (szállítási vállalat esetén). Az utóbbi esetben a szállító az árukért kapott díjat akarja maximalizálni. A többi esetben a cél nyilvánvaló. Legyen továbbá a súlykorlátozás mellett egy térfogati korlátozás is.

Az előző feladat jelölésein kívül még a következőket használjuk:

L a térfogati korlát,
b_j a j-edik tárgy térfogata,

Ekkor a feladat a következő:

$$(\max)\, z = \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=1}^{n} a_j x_j \leq K \tag{5}$$

$$\sum_{j=1}^{n} b_j x_j \leq L$$

$$x_j = 0 \text{ vagy } 1.$$

A feladat lényegesen nehezebb, mint az előző, hiszen még preferenciát sem tudunk megadni, minthogy ez teljesen különböző lehet az első, illetve második feltétel szerint. (Könnyű, nagy térfogatú, illetve nehéz, kis térfogatú tárgyak.)

A több-feltételes terhelési probléma egyébként $0-1$ lineáris egészértékű feladatra vezetett, sőt ha nem csak $1-1$ darab áll rendelkezésre belőlük és azok egyforma értékűek, akkor az általános lineáris tiszta egészértékű feladathoz jutottunk. Ugyanez a rendszer több különböző gyakorlati problémát ír le. A jelen fejezet célja azonban nem a matematikai hasonlóság keresése, hanem különböző alkalmazások bemutatása.

2.3 *Fix-költséges problémák*

A termelési költség általában két fő részből áll: egy fix-költségből és egy, a termelt mennyiséggel növekvő változó költségből. Most nem arról a fix-költségről van szó, amelyet minden termeléstől függetlenül ki kell fizetni (állóeszközök után fizetendő kamat, épületkarbantartás, állandó dolgozók alapbére stb.), mert hiszen ezt mindenképpen ki kell fizetni és a célfüggvényhez adott konstans nem változtatja meg az optimum helyét. Minden egyes sorozatgyártás beindításakor felmerül egy, a sorozatnagyságtól független költség. (Speciális szerszámok, betanulással kapcsolatos költségek stb.) Ez a költség azonban már nem független a termeléstől, mert egyáltalán nem merül fel, ha a megfelelő terméket nem állítjuk elő.

Nagy sorozatok gyártása esetén a darabszámot folytonosnak tekinthetjük. Az alábbi jelzéseket használjuk:

m az erőforrások száma,
n a gyártható termékek száma,
a_{ij} az i-edik erőforrásból a j-edik termék egységnyi mennyiségének előállításához szükséges mennyiség,
b_i az i-edik erőforrásból rendelkezésre álló mennyiség,
c_j a j-edik termék egységének előállításához szükséges költség (fix-költség nélkül),
d_j a j-edik termék termelésének beindításához szükséges fix-költség (egyelőre $d_j \geq 0$),
x_j a j-edik termékből gyártandó mennyiség,
δ_j értéke 1 vagy 0 attól függően, hogy a j-edik terméket gyártjuk, vagy nem.

Egyszerűség kedvéért feltételezzük, hogy a változó költség a termelt mennyiséggel arányos. A következő feladathoz jutunk:

$$\min \left(\sum_{j=1}^{n} c_j x_j + \sum_{j=1}^{n} \delta_j d_j \right)$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad (i = 1, 2, \ldots, m) \tag{6}$$

$$x_j \geq 0 \quad (j = 1, 2, \ldots, n)$$

$$\delta_j = \begin{cases} 0, & \text{ha } x_j = 0 \\ 1, & \text{ha } x_j > 0. \end{cases}$$

Az erőforrás korlátokon kívül szerepelhetnek természetesen bizonyos termékekre vagy azok kombinációjára vonatkozó alsó korlátok. Ezek a feltételek is átalakíthatók a fenti típusúvá. A c_j jelenthet hasznot is, ekkor $c_j < 0$. A $d_j \geq 0$ kikötést azért tettük, mert az ellenkező esetben lehet, hogy a feladatnak nincs minimuma, csak infimuma és a gazdasági értelmezés is problematikus. A $d_j < 0$ felfogható ugyan úgy, hogy a j-edik termék gyártásához népgazdasági érdek fűződik, ezért gyártásakor a gyár célprémiumot kap (minek összegéből már levontuk a gyártás beindításához szükséges költséget). Ha azonban a termék gyártása egyébként nem gazdaságos (a többiekhez viszonyítva), akkor az üzem minél kisebb mennyiséget kíván majd gyártani, ami nem is gazdaságos, a célprémium is hiábavaló, továbbá $x_j \to 0$ esetén a célfüggvény állandóan csökken, de $x_j = 0$ esetén hirtelen megnő. Ezt kiküszöbölendő, a következőképpen járhatunk el. Megtartjuk továbbra is a beindításhoz szükséges $d_j \geq 0$ költséget és a $p_j < 0$ prémiumot (a célfüggvény költséget minimalizáljuk, tehát a prémiumnak negatívnak kell lennie) csak akkor kapja meg az üzem, ha legalább g_j mennyiséget gyárt a szóban forgó termékből.

Bevezetve az

$$\varepsilon_j = \begin{cases} 0, & \text{ha } x_j < g_j \\ 1, & \text{ha } x_j \geq g_j \end{cases}$$

0—1 egészértékű változókat ($j = 1, 2, \ldots, n$) a célfüggvény a következőképpen alakul:

$$\min \sum_{j=1}^{n} (c_j x_j + \delta_j d_j + \varepsilon_j p_j),$$

a feltételek pedig változatlanok.

2.4 *Beruházási problémák*

A beruházások gazdaságosságának vizsgálatában fontos szerepet játszik az egészértékű programozás. Ha csak egyetlen beruházásról kellene dönteni, hogy végrehajtsuk-e vagy sem, akkor könnyű dolgunk volna, mert megoldanánk a feladatot azon feltételezéssel, hogy végrehajtjuk és azzal is, hogy nem hajtjuk végre. A probléma azonban általában nem így merül fel, hanem úgy, hogy egy tervet kell teljesíteni és ahhoz beruházások egész sorozatát kell végrehajtani.

Ezek közül bizonyosak helyettesíthetik egymást, mások kizárják egymást stb. Mindenképpen lehetetlen azonban az összes variációt kipróbálni ezek nagy számára való tekintettel (n lehetséges beruházás esetén 2^n), különösen azért, mert minden egyes variáció egy rendszerint elég bonyolult lineáris vagy nemlineáris feladat megoldását kívánja, melynek időigénye elég tekintélyes lehet. Tekintsük most a probléma egy egyszerű változatát, amelyben nincsenek folytonos változók és a beruházás egy jól meghatározott tevékenységet jelent adott erőforrás szükséglettel és költséggel. Tekintsük például az alábbi egyszerű modellt.

Legyen

c_j a j-edik beruházás költsége,
a_{ij} a j-edik beruházás szükséglete az i-edik erőforrásból,
b_i az i-edik erőforrásból rendelkezésre álló mennyiség,
d_{kj} a j-edik beruházás segítségével, a k-adik termék számára létrehozott új kapacitás,
h_k a k-adik termékből szükséges új kapacitás,
x_j *1* vagy *0* aszerint, hogy a j-edik beruházást végrehajtjuk-e vagy sem,
m az erőforrások száma,
n a beruházások száma,
r a gyártandó termékek száma.

Minimális összköltségű beruházás kombinációit keressük azok közül, amelyek együttesen nem lépik túl a rendelkezésre álló erőforrások mennyiségét és teljesítik az előírt termelési feladatot:

$$\min \sum_{j=1}^{n} c_j x_j$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad (i = 1, 2, \ldots, m) \tag{7}$$

$$\sum_{k=1}^{n} d_{kj} x_j \geq h_k \qquad (k = 1, 2, \ldots, r)$$

$$x_j = 0 \text{ vagy } 1.$$

A feltételek között szerepelhetnek a természeti adottságok, a körzetenkénti energiakorlátok, munkaerőkorlátozás stb. A költségeknél figyelembe lehet venni a beruházási költségen kívül az alapanyagok több időszakra vonatkozó diszkontált szállítási költségeinek összegét, ezzel is reálisabbá válik a modell változatlan szerkezet mellett.

Ha azonban az egyes létrehozandó új üzemek között is jelentős mennyiségű szállítások történnek, akkor ipartelepítési problémával állunk szemben, melyet a fenti modellel nem tudunk jól leírni. Erre a célra igen alkalmas az úgynevezett kvadratikus hozzárendelési feladat, mely szintén kizárólag *0—1* egészértékű változókat tartalmaz.

A feladat egy lehetséges megfogalmazása a következő: Tegyük fel egyszerűség kedvéért, hogy n számú, különböző típusú gyárat akarunk megépíteni, $q(\geq n)$ helyen úgy, hogy mindenütt legfeljebb egyet építhetünk.

c_{jt} a j-edik gyárnak a t-edik helyen való megvalósítási költsége,
f_{jp} a j-edik és p-edik gyár közötti szállítási intenzitás,
g_{ts} a t-edik helyről az s-edik helyre való egységnyi mennyiség szállítási költsége,
x_{jt} *1* vagy *0*, attól függően hogy a j-edik gyárat a t-edik helyen valósítjuk-e meg vagy sem.

Ekkor a feladat a következő:

$$\min \sum_{j=1}^{n} \sum_{t=1}^{q} \left(c_{jt} + \sum_{p=1}^{n} \sum_{s=1}^{q} f_{jt} g_{ts} x_{ps} \right) x_{jt}$$

$$\sum_{t=1}^{q} x_{jt} = 1 \qquad (j = 1, \ldots, n)$$

$$\sum_{j=1}^{n} x_{jt} \leq 1 \qquad (t = 1, \ldots, q)$$

$$x_{jt} = 0 \text{ vagy } 1.$$

3. A diszkrét programozás megoldási módszerei (Áttekintés)

Bár egy-egy speciális szerkezetű feladat megoldása néhány évvel korábban is történt, az első általános lineáris diszkrét programozási módszer GOMORY amerikai matematikus nevéhez fűződik és körülbelül egy évtizede született meg.

3.1 *A Gomory módszer*

Első változatát a szerző 1958-ban publikálta, majd a jelenlegi formájában 1963-ban a [19] cikk gyűjteményében (269—302 old.). A módszer lényege az, hogy először megoldjuk a megfelelő folytonos feladatot a szimplex módszer segítségével. Ha valamennyi változó, mely csak egész értéket vehet fel az eredeti feladatban a folytonos feladat optimális megoldásában is egészértékű, akkor megoldottuk a feladatot. Ellenkező esetben meghatározunk egy olyan egészértékű feltételt, amelyet az eredeti feladat valamennyi megengedett megoldása (azaz pl. tiszta feladat esetén a folytonos feladat megengedett tartományának valamennyi rácspontja) kielégít, de a jelenlegi (nem egészértékű) megoldás nem. Megoldjuk az újabb folytonos feladatot, amelyet úgy kapunk, hogy az eddigiekhez hozzávesszük a most meghatározott új feltételt. Az eljárást addig folytatjuk, míg az egészértékűnek deklarált változók a legutolsó folytonos feladat optimális megoldásában valamennyien valóban egész értékeket vesznek fel. Ez lesz az eredeti feladat optimális megoldása.

A módszer részletes leírására itt nincs lehetőség, de a Gomory-feltétel képzését egy példán keresztül szemléltetjük. Tekintsük az alábbi feladatot:

$$\begin{aligned} \max\ & 3x + y \\ & 4x - 4y \leq -1 \\ & 8x + 6y \leq 33 \\ & x \geq 0,\ y \geq 0 \\ & x, y \text{ egészek.} \end{aligned}$$

Egészítsük ki egyenlőségekké a fenti egyenlőtlenség formájában levő feltételeket, ekkor az eredetivel ekvivalens feladathoz jutunk, és minthogy a szereplő együtthatók és jobb oldalak egészek, az új bevezetett változókról is feltehetjük, hogy egészek.

$$\begin{aligned} \max\ & 3x + y \\ & 4x - 4y + u = -1 \\ & 8x + 6y + v = 33 \\ & x \geq 0,\quad y \geq 0,\quad u \geq 0,\quad v \geq 0. \\ & x, y, u, v \text{ egészek.} \end{aligned}$$

Szimplex módszer segítségével megoldva a folytonos változatot, az optimális megoldás

$$x = \frac{9}{4}, \qquad y = \frac{10}{4}, \qquad u = v = 0.$$

Kifejezve az x és y változókat:

$$x = \frac{9}{4} - \frac{3}{28}u - \frac{1}{14}v$$

$$y = \frac{5}{2} + \frac{1}{7}u + \frac{13}{14}v.$$

Az u, v nembázis változók megfelelő választásával minden megoldást megkapunk.

Válasszuk most külön az x kifejezésében az együtthatók egészrészét és törtrészét:

$$x = 2 - u - v + \frac{1}{4} + \frac{25}{28}u + \frac{13}{14}v.$$

Mivel az x változónak egész értéket kell felvennie, továbbá a $2 - u - v$ is egész, az

$$\frac{1}{4} + \frac{25}{28}u + \frac{13}{14}v$$

egész kell hogy legyen. Továbbá minthogy $u \geq 0$, $v \geq 0$, a kifejezés értéke nemnegatív egész. Minthogy azonban $u \geq 0$, $v \geq 0$ esetén ez a kifejezés nem lehet 0, tehát értéke legalább 1.

$$\frac{1}{4} + \frac{25}{28}u + \frac{13}{14}v \geq 1.$$

Egyenlőséggé kiegészítve az s pótváltozónak és azt kifejezve:

$$s = -\frac{3}{4} + \frac{25}{28}u + \frac{13}{14}v.$$

Az eddigiekből következik, hogy $s \geq 0$ és egész értékeket vehet csak fel. Ez a feltétel megfelel a kívánalmaknak, mert a folytonos feladat optimális megoldása ($u = v = 0$) nem elégíti ki, továbbá a fenti fejtegetésből nyilvánvaló, hogy viszont az eredeti feladat minden (egészértékű) megoldása (a tartomány minden rácspontja) kielégíti.

Gomory az említett dolgozatában bebizonyította, hogy — amennyiben a fenti módon az általa megadott sorrendben képezzük ezeket a pótlólagos felté-

teleket, az algoritmus véges sok lépésben végetér és megadja a feladat egészértékű optimális megoldását, amennyiben létezik a feltételeket kielégítő (tehát egészértékű) megoldás.[2]

GOMORY ugyanebben a dolgozatában kidolgozta a módszernek vegyes egészértékű feladatokra alkalmas módosítását.

Megjegyzendő, hogy nem kell minden egyes újabb feltétel hozzávételekor a megfelelő folytonos feladatot az elejétől megoldani, mert az ún. duál szimplex módszer lehetővé teszi, hogy az algoritmust az előző feladat optimális megoldásából kiindulva folytassuk.

A módszer eredeti formájában nem túl nagy méretű feladatok esetén (20—30 változó és ugyanennyi egyenlőtlenség alakú feltétel) is már elég lassú. Sok kísérlet történt az algoritmus gyorsítására, például jó eredményekről számol be MARTIN [33] dolgozatában.

Gomory-típusú módszer, ill. a számítási tapasztalatok találhatók az alábbi dolgozatokban: DALTON és LLEWELLYN [12], GLOVER [16], WILSON [42].

3.2 *Megoldás konvex programozással*

Egyszerűség kedvéért tekintsük az (1) lineáris kevert egészértékű feladatot abban az esetben, amikor valamennyi szereplő együttható egész és $r_j = 1$ $(j = 1, 2, \ldots, k)$. Mint korábban láttuk, az $r_j > 1$ visszavezethető erre az esetre. A feladat tehát röviden a következő formában foglalható össze:

$$\min \mathbf{c}^T\mathbf{x} \tag{8}$$

$$\mathbf{Ax} \geq \mathbf{b} \tag{9}$$

$$x_j = 0 \text{ vagy } 1 \qquad (j = 1, 2, \ldots, k) \tag{10}$$

$$x_j \geq 0 \qquad (j = \kappa + 1, \ldots, m). \tag{11}$$

Tűzzük ki először célul, hogy egy megengedett megoldást, azaz a (9)—(11) feltételeket kielégítő $\mathbf{x}$ vektort keressük. Ebből a célból tekintsük az alábbi feladatot:

$$\min \sum_{j=1}^{k} x_j(1 - x_j) \tag{12}$$

$$\mathbf{Ax} \geq \mathbf{b} \tag{13}$$

$$0 \leq x_j \leq 1 \qquad (j = 1, 2, \ldots, k) \tag{14}$$

$$x_j \geq 0 \qquad (j = k + 1, \ldots, m). \tag{15}$$

Ez már egy folytonos feladat, azonban a célfüggvény nem lineáris, hanem konkáv. (Elvégezve a beszorzásokat egy lineáris kifejezés és egy negatív definit kvadratikus alak összegét kapjuk.)

[2] Gomory előtt többen kísérleteztek ilyen vágásokkal, azonban a megfelelő algoritmusok konvergenciáját nem tudták igazolni, sőt a legtöbb esetben olyan példákat is tudtak konstruálni, amelyekre a módszer nem adta meg az optimális megoldást.

Ha ezt a feladatot meg tudjuk oldani, akkor három eset lehetséges.

a) Nincs megengedett megoldás. Ekkor az eredeti feladatnak sincs, hiszen a (14) feltétel gyengébb, mint a (10), a másik kettő pedig azonos.

b) Az optimális megoldáshoz tartozó célfüggvény-érték pozitív. A (9)—(11) feltételeket kielégítő megoldás ekkor sem létezik, mert ha volna ilyen, az kielégítené a (13)—(15) feltételeket is és *0* célfüggvény-értéket szolgáltatna.

c) Az optimális megoldáshoz tartozó célfüggvény-érték *0*. Ekkor nyilvánvalóan a (9)—(11) feltételeket kielégítő megoldáshoz jutunk, hiszen ekkor az összeadandók nemnegativitásából következik, hogy

$$x_j(1 - x_j) = 0 \qquad (j = 1 \ldots, k),$$

azaz

$$x_j = 0 \text{ vagy } 1.$$

Következő lépésként, ha a *c)* esethez jutottunk, az eddiginél jobb megengedett megoldásokat keresünk. Jelöljük általában az r-edik lépésben kapott megengedett megoldást $\mathbf{x}^{(r)}$-rel. Ekkor a (12)—(15) feladathoz csatoljuk a

$$\sum_{j=1}^{k} c_j x_j \leq \sum_{j=1}^{k} c_j x_j^{(r)} - 1 \tag{16}$$

feltételt, ahol a jobb oldalon egy konstans áll, minthogy $\mathbf{x}^{(r)}$-et már meghatároztuk.

Addig folytatjuk az eljárást, míg az utolsó lépésben az *a)* vagy *b)* esetet kapjuk, azaz az eredeti, (8)—(11) feladatnak nincs a legutoljára kapottnál jobb megengedett megoldása, ez tehát az optimális megoldás.

Természetesen lehet nagyobb lépésközzel is haladni, azaz a (16) egyenlőtlenségben 1-nél többet levonni, ekkor azonban az *a)* vagy *b)* eset elérésekor az addigi legjobb célfüggvény-érték és a csökkentett érték közötti részt is fel kell kutatni (például a szokásos intervallum felezéssel).

Most térjünk vissza a (12)—(15), ill. a (12)—(16) feladat megoldására. Ha egy konkáv függvényt akarunk minimalizálni egy konvex poliéder felett, akkor lokális, azaz helyi szélsőértékekhez is juthatunk. Amennyiben tehát a minimum *0*, akkor elértük a globális minimumot, mert a (12) célfüggvény ennél kisebb értéket nem vehet fel. Ha az *a)* eset fordul elő, akkor is minden rendben van. Ha azonban a *b)* esethez jutunk, akkor lehetséges, hogy csak egy lokális minimumot kaptunk és van ennél kisebb célfüggvény-érték is. Közelítő módszerhez juthatunk a Monte Carlo-módszer alkalmazásával, vagy úgy hogy a (13)—(16) poliéder több csúcsából kiindulva oldjuk meg a feladatot.

Rutledge [40] dolgozatában azonban másképp oldotta meg ezt a problémát. Helyettesítsük a (12) célfüggvényt a következővel:

$$\max \sum_{j=1}^{k} \left(x_j - \frac{1}{2}\right)^2. \tag{17}$$

A *0* minimum szerepét most a

$$\sum_{j=1}^{k} \left(\frac{1}{2}\right)^2 = \frac{k}{4} \tag{18}$$

maximum veszi át, egyébként minden változatlan. Sajnos ebben az esetben is lokális szélsőértékekkel találkozunk, mert most egy konvex függvény maximumát keressük egy konvex tartomány felett. RUTLEDGE említett dolgozatában azonban bebizonyította, hogy ha a

$$\frac{\sum_{j=1}^{k}\left(x_j-\frac{1}{2}\right)^2+S}{-\sum_{j=1}^{n}c_jx_j+Q} \tag{19}$$

függvényt maximalizáljuk, megfelelő S és Q esetén a (19) függvény (13)—(15) tartományon vett maximum helye a (8)—(11) feladat optimális megoldását szolgáltatja, azaz a lokális optimum egyben globális is. Módszert is adott ilyen Q és S számok meghatározására és a (19) célfüggvény maximalizálására a (13)—(15) feltételek mellett.

3.3 *Leszámlálási algoritmusok*

A [25] jegyzetben kombinatorikus módszerek összefoglaló cím alatt az olvasó részletesen kidolgozva megtalál néhány leszámlálási módszert, továbbá a következő szakaszban tárgyalt korlátozás és szétválasztás módszerét és a dinamikus programozás alkalmazását diszkrét feladatok megoldására.

Foglalkozzunk egyelőre a *0—1* tiszta diszkrét feladattal. A leszámlálási algoritmusok lényege, hogy valamilyen képzeletbeli sorrendbe állítja a szóba jövő rácspontokat, azaz n változó esetén 2^n vektort. Valamennyi vektor kipróbálása nem lehetséges, hiszen már $n = 30$ esetén $2^{30} = 10^9$, ami másodpercenként 1000 behelyettesítést véve néhány száz gépórát jelent. Ehelyett a leszámlálási algoritmusok a fenti képzeletbeli sorban nagy ugrásokkal haladnak előre, valamilyen módon biztosítva, hogy a közben kihagyott változó értékek vagy nem megengedett megoldásai a feladatnak, vagy az addig talált legjobb megoldásnál nem szolgáltatnak jobbat.

A legtöbb leszámlálási eljárásnál lehetőség van arra is, hogy a megoldás során nyert információknak megfelelően csoportosítsuk a még meg nem vizsgált vektorokat, és így hamarabb jussunk megengedett megoldáshoz, ill. az eddigi legjobbnál jobbhoz. Ez viszont ismét a hátralevő vektorhalmaz egy részének elhagyhatóságát jelenti. Az utóbbi csoportba tartozik például BALAS [1] és [2] dolgozata, valamint GLOVER [17] munkája. Hasonló típusú módszerekkel, illetve ezek értékelésével gépi felhasználás szempontjából sokan foglalkoztak, például GLOVER és ZIONTS [18], FLEISCHMANN [14], GEOFFRION [15] stb.

Arra az esetre, amikor a képzeletbeli sorrendet nem változtatjuk, példa LAWLER és BELL [28] algoritmusa. Hogy az olvasó képet nyerjen a leszámlálási algoritmusokról, ezt a módszert egyszerűsített formában bemutatjuk. A lineáris, *0—1* tiszta diszkrét probléma mindig átírható a következő alakúvá:

$$\min \sum_{j=1}^{n} c_jx_j$$

$$\sum_{j=1}^{n} a_{ij}x_j - \sum_{j=1}^{n} d_{ij}x_j \geqq b_i \qquad (i = 1, \ldots, m) \tag{20}$$

$$x_j = 0 \text{ vagy } 1 \qquad (j = 1, 2, \ldots, n) \tag{21}$$

m feltétel és n változó van, a szereplő mátrixok és vektorok a megfelelő méretűek, továbbá

$$c \geqq 0, \quad a_{ij} \geqq 0, \quad d_{ij} \geqq 0 \qquad (i = 1, \ldots, m; \quad j = 1, \ldots, n)$$

Nem jelent megszorítást, ha feltesszük, hogy az a_{ij} és d_{ij} együtthatók közül legfeljebb az egyik nullától különböző. A $0-1$ komponensű n dimenziós vektorok képzeletbeli sorbaállítása (ebben a módszerben egyszer és mindenkorra) úgy történik, hogy a változók értékeit egymás mellé írva az így kapott számot bináris (kettes számrendszerbeli) számnak képzelve növekvő sorrendben írjuk fel őket. Például $n = 3$ esetén:

$$000,\ 001,\ 010,\ 011,\ 100,\ 101,\ 110,\ 111.$$

Definiáljuk továbbá egy tetszőleges n dimenziós $0-1$ komponensű $\mathbf{x}$ vektorhoz az $\mathbf{x}^*$-ot úgy, hogy legyen

$$\mathbf{x}^* \geqq \mathbf{x}$$

és az $\mathbf{x}$ után következők közül $\mathbf{x}^*$ az első ilyen. Legyen továbbá $\mathbf{x}^{*-}$ az $\mathbf{x}^*$-ot közvetlenül megelőző vektor. Példák $n = 5$ esetén

$\mathbf{x} = (0, 0, 1, 0, 0)$	$\mathbf{x} = (0, 1, 1, 0, 0)$	$\mathbf{x} = (1, 1, 1, 0, 0)$
$\mathbf{x}^{*-} = (0, 0, 1, 1, 1)$	$\mathbf{x}^{*-} = (0, 1, 1, 1, 1)$	$\mathbf{x}^{*-} = (1, 1, 1, 1, 1)$
$\mathbf{x}^* = (0, 1, 0, 0, 0)$	$\mathbf{x}^* = (1, 0, 0, 0, 0)$	

Az $\mathbf{x}^{*-}$ definiálható az $\mathbf{x}^*$-tól függetlenül is:

$$\mathbf{x}^{*-} \geqq \mathbf{x}$$

és $\mathbf{x}^{*-}$ az utolsó ilyen tulajdonságú az $\mathbf{x}$ után következők közül. A fenti utolsó példában $\mathbf{x}^*$ nem létezik.

Könnyen belátható, hogy a leszámlálásban $\mathbf{x}$-ről $\mathbf{x}^*$-ra ugorhatunk (azaz $\mathbf{x}$ és $\mathbf{x}^*$ között nincs megengedett és az eddigi legjobbnál jobb megoldás, ha a következő feltételek valamelyike teljesül. (Jelölje az $\mathbf{x}$-ig talált legjobb megoldást $\hat{\mathbf{x}}$):

a) ha $\mathbf{c}^T\mathbf{x} \geqq \mathbf{c}^T\hat{\mathbf{x}}$,
b) ha $\mathbf{x}$ megengedett megoldás, azaz kielégíti a (20) feltételeket,
c) ha bármely i-re kielégíti a

$$\sum_{j=1}^{n} a_{ij}x_j^{*-} - \sum_{j=1}^{n} d_{ij}x_j < b_i$$

egyenlőtlenséget.

A módszer tehát abban áll, hogy a fenti sorrendben vizsgáljuk egymás után az n dimenziós $0-1$ komponensű vektorokat. Ha valamelyikre teljesül az *a)*, *b)* vagy *c)* feltétel, akkor az $\mathbf{x}^*$ vektornál egyébként a következőnél foly-

tatjuk a leszámlálást. A változók jó rendezése lényeges! A módszer csekély változtatással abban az esetben is használható, ha a célfüggvény tetszőleges monoton nemcsökkenő függvény, az egyenlőtlenségek baloldalán álló függvények pedig két monoton nemcsökkenő függvény különbségeként állíthatók elő.

A korábban említett leszámlálási algoritmusok a most vázolt eljárástól nemcsak abban különböznek, hogy a leszámlálás sorrendje nem előre rögzített, hanem még két további fontos jellemvonásuk van.

A feltételek egy jól megválasztott nemnegatív lineáris kombinációját veszszük és először ezen végezzük el az ellenőrzést. Ha ugyanis már ezt sem elégíti ki egy n dimenziós vektor, akkor a teljes rendszert annál kevésbé. Az is nyilvánvalóvá válhat már ezen egyetlen feltételből, hogy bizonyos esetekben egyes változókat *0*, másokat *1* értékűnek kell választani. Az említett nemnegatív szorzók megválasztása igen lényeges, az irodalomban részletesen foglalkoznak vele.

A másik jellemvonás az ún. tesztek alkalmazása. Tesztnek nevezünk egy olyan kritériumot, amely segítségével könnyen eldönthető bizonyos feltételek teljesülése esetén, hogy a vizsgált megoldás nem elégítheti ki az összes feltételt vagy nem lehet jobb, mint az addig talált legjobb. Teszteket az eredeti feltételekre és az említett — a feltételekből nemnegatív lineáris kombinációként kapott — pótlólagos feltételre is építhetünk.

3.4 *A korlátozás és szétválasztás módszere*

A korlátozás és szátválasztás módszerét először Land és Doig [26] alkalmazta vegyes egészértékű feladatokra. Az utazó ügynök probléma megoldására Little és mások [30] alkalmazták a módszert. Azóta igen sokan különböző típusú elméleti és gyakorlati feladatokra sikerrel dolgozták ki a módszer variánsait. Ugyanis az alapgondolatok egyszerűen leírhatók, azonban hatékony algoritmushoz csak úgy jutunk, hogy ha a szereplő kritériumokat a feladat természetének megfelelően választjuk ki. A korlátozás és szétválasztás módszeréről részletes ismertetést és bibliográfiát talál az olvasó Lawler és Wood [29] dolgozatában. A módszer lényegét — egyszerűség kedvéért *0—1* tiszta egészértékű feladatra — a következőképpen lehet összefoglalni:

A megengedett megoldások halmazát — például egyes változók *0*, ill. *1* szinten való rögzítésével — több részhalmazra bontjuk. Minden halmazhoz kiszámítjuk valamilyen alkalmas módon (minimalizálás esetén) a célfüggvény egy jó alsó becslését. Kiválasztjuk azt a halmazt, ahol ez a becslés a legalacsonyabb értéket adja és ott folytatjuk a felbontást. A felbontás után újra alsó becslések következnek, majd ismét a minimális alsó becslésű halmaz megkeresése stb. A minimális alsó becslésnél mindig a még felbontatlan halmazokat vesszük csak figyelembe, hiszen a felbontott halmazoknak már a részhalmazait is vizsgáljuk. A módszer sikere azon múlik, hogyan választjuk meg a felbontási kritériumot, továbbá milyen módon határozzuk meg az egyes halmazokban a célfüggvény alsó becslését. Ha például az alsó becslés nem jó, előfordul, hogy egy megoldáshalmazt csak azért bontunk tovább, mert a hozzátartozó célfüggvény-érték alsó becslés rossz (túl alacsony) és nem azért, mert a halmazban levő megoldások némelyikéhez valóban alacsony célfüggvény-érték tartozik.

Illusztratív példaként tekintsük a következő egyszerű típusú, de mégis fontos feladatot:

$$\min c_1x_1 + c_2x_2 + \ldots + c_nx_n \tag{22}$$

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \geq b \tag{23}$$

$$x_j = 0 \text{ vagy } 1 \qquad (j = 1, 2, \ldots, n). \tag{24}$$

Tegyük fel, hogy a változók már úgy vannak rendezve, hogy

$$\frac{c_1}{a_1} \leq \frac{c_2}{a_2} \leq \ldots \leq \frac{c_n}{a_n}. \tag{25}$$

Könnyen belátható, hogy amennyiben véletlenül

$$a_1 + a_2 + \ldots + a_p = b \tag{26}$$

valamilyen $1 \leq p \leq n$ esetén, akkor az optimális megoldás

$$x_1 = \ldots = x_p = 1 \qquad x_{p+1} = \ldots = x_n = 0. \tag{27}$$

Ha azonban

$$\sum_{j=1}^{p} a_j < b < \sum_{j=1}^{p+1} a_j \tag{28}$$

valamilyen $0 \leq p < n$ esetén, akkor előfordulhat, hogy a (27) egyenlőségekben szereplő megoldás nem optimális, sőt nem is állítható úgy elő, hogy csupán néhány 0 értékű változó 1 lesz, hanem ettől teljesen különböző megoldást kapunk. Ebben a feladatban a korlátozás és szétválasztás módszerét például úgy alkalmazhatjuk, hogy egy — valamilyen kritérium szerint (pl. a (25) rendezésnek megfelelő sorrendben az első még nem rögzített változó) — kiválasztott változó értékét 0, illetve 1 szinten rögzítjük. Ilyen módon két feladathoz jutunk, amelyben csak $n-1$ változó szerepel. Az egyik megoldáshalmaz az $x_1 = 0$, másik az $x_1 = 1$ változó értékrögzítésnek felel meg. A többi változó értéke mindkét halmazban tetszőleges. Az alsó korlát becslést pedig úgy kapjuk, hogy az $x_1 = 0$, ill. $x_1 = 1$ értékadással előállított feladat folytonos változatát oldjuk meg, azaz a

$$\begin{gathered} \min c_2x_2 + \ldots + c_nx_n \\ a_2x_2 + \ldots + a_nx_n \geq b \\ 0 \leq x_j \leq 1 \qquad (j = 2, \ldots, n), \end{gathered} \tag{29}$$

illetve a

$$\begin{gathered} \min c_1 + c_2x_2 + \ldots + c_nx_n \\ a_2x_2 + \ldots + a_nx_n \geq b - a_1 \\ 0 \leq x_j \leq 1 \qquad (j = 2, \ldots, n) \end{gathered} \tag{29}$$

feladatokat. Mivel az $x_j = 0$ vagy 1 helyett a gyengébb $0 \leq x_j \leq 1$ feltételt tekintettük, azaz bővebb tartományon optimalizálunk, a minimum kisebb vagy egyenlő, mint az egészértékű minimum. Így tehát a (29) és (30) feladatok megoldásával alsó korláthoz jutunk az $x_1 = 0$, ill. $x_1 = 1$ rögzítéssel nyert részhalmazokra vonatkozóan. Az is látszik, hogy ez a becslés elég jó. Másrészt a (29) és (30) feladatok megoldása igen egyszerű. Könnyen belátható, hogy — tekintettel a (25) rendezésre — a (29) feladat optimális megoldása:

$$x_2 = \ldots = x_p = 1\,; \qquad x_{p+1} = \frac{1}{a_{p+1}}\left(b - \sum_{j=1}^{p} a_j\right); \qquad x_{p+2} = \ldots = x_n = 0\,,$$

ahol

$$\sum_{j=2}^{p} a_j \leq b < \sum_{j=2}^{p+1} a_j\,.$$

Hasonlóan kapjuk a (30) feladat megoldását:

$$x_2 = \ldots = x_q = 1\,; \qquad x_{q+1} = \frac{1}{a_{q+1}}\left(b - \sum_{j=1}^{p} a_j\right); \qquad x_{q+1} = \ldots = x_n = 0\,,$$

ahol

$$\sum_{j=1}^{q} a_j \leq b < \sum_{j=1}^{q+1} a_j\,.$$

A felbontást abban a csoportban folytatjuk, amelyben a célfüggvény-érték becslése alacsonyabb, azaz az $x_1 = 0$, ill. $x_1 = 1$ csoportban aszerint, hogy a

$$\sum_{j=2}^{p+1} c_j x_j \quad \text{vagy a} \quad \sum_{j=1}^{q+1} c_j x_j$$

mennyiség kisebb. Egyenlőség esetén közömbös, hogy melyik halmazt választjuk. A felbontást addig folytatjuk, míg valamelyik halmazban csak egyetlen megengedett megoldás van és a hozzá tartozó célfüggvény-érték nem nagyobb, mint a többi, még felbontatlan megoldáshalmazhoz tartozó alsó becslés. Ekkor megkaptuk a feladat optimális megoldását.

A használt felbontás akkor jó, ha úgy bontja fel a megoldásokat, hogy az alsó becslések nagyon különböznek egymástól, feltéve természetesen, hogy az alsó becslésre jó módszerünk van és nem vagyunk nagyon távol a szóban forgó részhalmazban elért valódi minimumtól. Különösen jó a felbontás, ha az alacsony alsó becslésű halmaz sokkal kevesebb megoldást tartalmaz. A módszer alkalmazása során fontos olyan kritériumokat is találnunk, amivel felismerhetjük, hogy egy megoldáshalmaz elemei nem elégítik ki a feltételeket, így ezt a részhalmazt a továbbiakban figyelmen kívül hagyhatjuk. (Itt a fenti feladatnál bonyolultabbakra gondolunk.) A kritériumok és az alsó becslés módszerének megválasztásán múlik a módszer sikeres alkalmazása.

3.5 *Vegyes egészértékű problémák*

Az előző két szakasz módszereit csak tiszta egészértékű feladatokra fogalmaztuk meg. Mivel a Gomory-módszer is és a 3.2 szakaszban leírt módszer is alapjában folytonos módszerek, itt nem okozott nehézséget a vegyes feladatok kezelése.

Mind a leszámlálási algoritmusok esetében, mind a korlátozás és szétválasztás típusú módszereknél elvileg elképzelhető, hogy az egészértékű változók felmerülő értékeire — egyszerű behelyettesítés helyett — egy tiszta folytonos feladatot oldunk meg. Ha a folytonos változók száma nem nagy vagy a feladat speciális szerkezete miatt a fellépő folytonos feladatok gyorsan megoldhatók, akkor ez járható út. Ellenkező esetben azonban valami fajta dekompozícióra van szükségünk. Benders [10] dolgozatában leír egy ilyen módszert, amely egy vegyes egészértékű feladat megoldását visszavezeti egy tiszta egészértékű és egy tiszta folytonos feladat megoldására. Minthogy ebben egy konvex poliéder összes csúcsainak előállítása is szerepel, nagyobb feladatok esetén ebben a formában nem megvalósítható. Azonban a szerző a módszer egy módosítását is megadja, mely elkerüli ezt a nehézséget és egy véges sok lépésből álló iteratív eljárást szolgáltat, melyben tiszta egészértékű és tiszta folytonos feladatok sorozatát kell megoldani. Bármely tiszta egészértékű feladat megoldására alkalmas módszer használható.

3.6 *Közelítő megoldások*

Bizonyos esetekben a feladat nagy mérete vagy a rendelkezésre álló kevés idő miatt közelítő megoldásokat használunk. Ez annyit jelent, hogy a végül elfogadott megoldásról nem tudjuk biztosan, hogy optimális, csak annyit, hogy bizonyos értelemben jó megoldás. A legtöbb ilyen módszer egy speciális feladatra készül, mint például Baumol és Kuhn [6] munkája. Van azonban olyan törekvés is, hogy legalább — gyengébb vagy erősebb értelemben — lokális optimumokhoz jussunk egy tetszőleges egészértékű feladat esetén. Ilyen módszer található például Reiter és Sherman [39] elméleti igényű munkájában. Lokálisan optimálisnak nevezünk egy megoldást, ha környezetében nincs nála jobb megoldás. A környezet definíciójától függően kapunk sokatmondó, de nagyobb számítási igényű vagy gyengébb állítást tartalmazó, de gyorsabban számítható algoritmusokat.

(Beérkezett: 1969. III. 10.)

IRODALOM

[1] Balas, E.: An Additive Algorithm for Solving Linear Programs with Zero-One Variables. Operations Research, Vol. 13, 1965 517—546 p.

[2] Balas, E.: Discrete Programming by the Filter Method. Opers. Res. Vol. 15. 1967. 5., 915—957 p.

[3] Balinski, M. L.: Fixed-Cost Transportation Problems. Naval Research Logistics Quarterly, Vol. 8. 1961. 5136—5139 p.

[4] Balinski, M. L.: Integer Programming: Methods, Uses, Computation. Management Science, Vol. 12. 1965, 253—313 p.

[5] Balinski, M. L.—Quart, R. E.: On an Integer Program for a Delivery Problem. Operations Research, Vol. 12. 1964, 300—304 p.

[6] Baumol, W. J.—Kuhn, H. W.: An Approximate Algorithm for the Fixed-Charges Transportation Problem. Naval Research Logistics Quarterly, Vol. 9. 1962, 1—15 p.
[7] Bellman, R. E.: Dynamic Programming. Princeton University Press, Princeton, 1957.
[8] Bellman, R. E.—Dreyfus, S. E.: Applied Dynamic Programming, Princeton University Press, Princeton, 1962.
[9] Bellmore, M.—Neuhauser, G. L.: The Travelling Salesman Problem. A Survay. Operations Research, 16. 1968, 538—558 p.
[10] Benders, J. F.: Partitioning Procedures for Solving Mixed-Variables Programming Problems, Numerische Mathematik, Vol. 4. 1962, 238—252 p.
[11] Charnes, A.—Cooper, W. W.: Management Models and Industrial Applications of Linear Programming. John Wiley, New York, 1961.
[12] Dalton, R. E.—Llewellyn, R. W.: An Extension of the Gomory Mixed-Integer Algorithm to Mixed-Discrete Variables, Management Science, Vol. 12, Series A. 1967, 569—575 p.
[13] Dantzig, G. B.: Linear Programming and Extensions, Princeton University Press, Princeton, N. J., 1963.
[14] Fleischmann, B.: Computational Experience with the Algorithm of Balas. Operations Research, Vol. 15. 1967. 153—155 p.
[15] Geoffrion, A.: Integer Programming by Implicit Enumeration and Balas's Method, SIAM Review, Vol. 9. 1967, 178—190 p.
[16] Glover, F.: A Bound Escalation Method for the Solution of Integer Linear Programs. Cahiers du Centre d'Études de Recherche Operationnelle, Vol. 8. 1965, 131—168 p.
[17] Glover, F.: A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem. Operations Research Vol. 13. 1965, 879—919 p.
[18] Glover, F.—Zionts, S.: A Note on the Additive Algorithm of Balas. Operations Research, Vol. 13. 546—549 p.
[19] Graves, R. L.—Wolfe, P. Editors: Recent Advances in Mathematical Programming. McGraw-Hill Book Company, Inc., 1963.
[20] Hadley, G.: Linear Programming, Addison-Wesley, Reading, Massachusetts, 1962.
[21] Hadley, G.: Nonlinear and Dynamic Programming. Addison-Wesley, Reading. Massachusetts, 1964.
[22] Ignall, E.—Schrage, L.: Application of the Branch and Bound Technique to Some Flow-Shop Scheduling Problems, Operations Research, Vol. 13. 1965, 400—412 p.
[23] Koopmans, Tjalling C.—Beckman, M.: Assignment Problems and the Location of Economic Activities. Econometrica. Vol. 25. 1967, 53—76 p.
[24] Kovács, L. B.: A diszkrét programozás kombinatorikus módszerei. Az operációkutatás matematikai módszerei c. tanfolyam jegyzete. Bólyai János Matematikai Társulat, Budapest. (Sajtó alatt.)
[25] Kovács, L. B.: Leszámlálási struktúrák és alkalmazásuk diszkrét programozási feladatok megoldására. Matemətikai Lapok, XIX. 1968. 1—2. 33—48 p.
[26] Land, A. H.—Doig, A. G.: An automatic Method of Solving Discrete Programming Problems, Econometrica, Vol. 28. 1960, 497—520 p.
[27] Lawler, E. L.: The Quadratic Assignment Problem, Management Science, Vol. 9. 1963, 586—599 p.
[28] Lawler, E. L.—Bell, M. D.: A Method for Solving Discrete Optimization Problems. Operations Research, Vol. 14. 1966, 1098—1112 p.
[29] Lawler, E. L.—Wood, D. E.: Branch-and-Bound Methods: A Survey Operations Research, 14. 1966, 699—719 p.
[30] Little, J. D. C.—Murty, K. G.—Sweeney, D. W.—Caroline, K.: An Algorithm for the Travelling Salesman Problem. Operations Research, Vol. 11. 1963, 972—989 p.
[31] Lommicki, Z. A.: A Branch and Bound Algorithm for the Exact Solution of the Three-Machine Scheduling Problem, Operational Research Quarterly, Vol. 16. 1965, 89—100 p.
[32] McCluskey, E. J.: Minimization of Boolean Functions. Bell System Technical Journal, Vol. XXXVIII, 1959, 1485—1512 p.
[33] Martin, G. T.: An Accelerated Euclidean Algorithm for Integer Linear Programming, 311—318 p. in [19].
[34] Neuhauser, G. L.: Introduction to Dynamic Programming. Wiley, New York, 1966.

[35] PETERSEN, C. C.: Computational Experience with Variants of the Balas Algorithm Applied to the Selection of R and D projects. Management Science, Vol. 13. 1967, 736—750 p.
[36] PRÉKOPA, A.: Lineáris programozás I. Az operációkutatás matematikai módszerei c. tanfolyam jegyzete. Bólyai János Matematikai Társulat, Budapest, 1968.
[37] PRÉKOPA, A.: On the Probability Distribution of the Optimum of a Random Linear Program. SIAM Journal on Control 4. 1966, 1. 211—222 p.
[38] QUINE, W. V.: A Way to Simplify Truth Functions. American Mathematical Monthly, Vol. 62. 1955. 627—631 p.
[39] REITER, S.—SHERMAN, G.: Discrete Optimizing Journal of the Society for Industrial and Applied Mathematics, Vol. 3. 1965, 864—889 p.
[40] RUTLEDGE, R. W.: A Simplex Method for Zero-One Mixed Integer Linear Programs, Journal of Mathematical Analysis and Applications, Vol. 18. 1967, 377—390 p.
[41] WAGNER, H. M.—GIGLIO—RICHARD, J.—GLASER, GEORGE, R.: Preventive Maintenance Scheduling by Mathematical Programming, Management Science, Vol. 10. 1964, 316—334 p.
[42] WILSON, R. S.: Stronger Cuts in Gomory's All-Integer Programmi Algorithm. Operations Research, Vol. 15 1967, 155—157 p.
[43] YOUNG, R. D.: A Simplified (All-Integer) Integer Programming Algorithm. Operations Research. 16 1968. 4. 750—782 p.
[44] BOD, P.: Újabb eredmények az egészértékű lineáris programozásban. Operációkutatás Aktuális Problémái sorozat. 6. sz. MTESZ Automatizálási, Információfeldolgozási és Operációkutatási Tanács kiad. 1967. 50 p.