FOGALMAK ÉS MÓDSZEREK

SIVÁK JÓZSEF

Nagyméretű feladatok megoldási módszerei

Az elmúlt évek során rendkívül gazdag irodalmi anyag halmozódott föl a nagyméretű programozási feladatok megoldási módszereiről. G. Dantzig és P. Wolfe 1960-ban publikálták gondolatébresztő tanulmányukat a lineáris programozási feladatok megoldásának dekompozíciós elvéről [5]. Az azóta eltelt években sok új javaslat látott napvilágot. Az új megoldási lehetőségek keresését a rendelkezésre álló számítógépek kis kapacitásából származó nehézségek áthidalása tette szükségessé. Felvetődött ugyanis az egyre nagyobb méretű feladatok megoldásának igénye. Így születtek meg a különböző dekompozíciós algoritmusok.

A számítógéptechnika gyors fejlődése — a harmadik generációs gépek megjelenése — révén lehetővé vált, hogy már több ezer feltételes programozási feladatokat is felbontás nélkül megoldjanak. Az új megoldási lehetőségek keresésének indítékát ezúttal a számítások idő- és költségigényének csökkentése adta.

Meg kell mondanunk, hogy az eredeti várakozások teljesülését nem mondhatjuk egyértelműen jónak, az új módszerek nem mindig hozták meg a remélt sikert.

A felhalmozódott ismeretanyag, a körülbelül 60-féle új eljárás három szempontból vált hasznossá. Először hozzájárult a számítástechnikai fogások eszköztárának bővítéséhez, másrészt olyan megoldási fogások kifejlesztéséhez, amelyek gyümölcsözőek az eddig nehezen kezelhető feladatok megoldásában (pl. nemlineáris feladatok megoldása). Harmadsorban a javasolt eljárások egy részének olyan közgazdasági interpretáció adható, amivel lehetővé válik bizonyos közgazdasági — elsősorban tervezési — folyamatok absztrakt modellezése, elemzése is.

Ebben a cikkben kísérletet teszünk arra, hogy a rendelkezésre álló irodalom alapján áttekintést adjunk a nagyméretű feladatok megoldási módszereinek kutatásában eddig elért eredményekről, valamint utaljunk a további kutatási irányokra. A cikk célja egyrészt az, hogy a témakörben kevésbé jártas érdeklődőknek új ismereteket adjon, másfelől az irodalmi anyag bizonyos fokú egységesítésével az avatottabbaknak rendszerezési szempontokat alakítsunk ki. Valamely témakör egységesítése sok esetben erőszakolt egyszerűsítésekhez vezet, amit ebben az esetben sem kerülhettünk el.

Az összefoglalás szempontjainak kialakításában a témakör klasszikusának számító G. Dantzig néhány tanulmánya mellett elsősorban A. Geoffrion összefoglaló cikkére támaszkodtunk [6], [7a] és [11].

A cikk három részből áll. Az első részben a nagyméretű feladatok néhány jellegzetes típusáról írunk. A második részben tárgyaljuk a megoldási mód-

szereket. A tárgyalásban kitérünk a szimplex módszer továbbfejlesztését felhasználó módszerekre, majd az új megoldási algoritmusok főbb közös tulajdonságaira. A harmadik rész a dekompozíciós algoritmusok egy lehetséges közgazdasági interpretációjával foglalkozik. Az áttekintésben el akartuk kerülni a túlzott részletezettséget. Alapvetően azt tekintettük célunknak, hogy arra hívjuk fel a figyelmet, milyen módon kezelhetők az egyes algoritmusok. A részletek mellett felhívjuk a figyelmet az eredeti pulikációkra.

1. A nagyméretű feladatokról

Ha valamilyen definíciót akarnánk adni arra, hogy milyen mérettől kezdve nevezünk egy programozási feladatot nagyméretűnek, akkor bizonyára nehézségeink lennének. A nagy méret önmagában viszonylagos fogalom. A gyakorlatban a nagy méretet sokszor a feladat méretének és a rendelkezésre álló számológép kapacitásának viszonyával fejezzük ki.

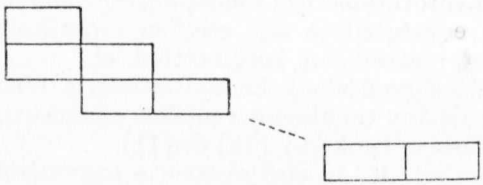
A nagyméretű feladatok megoldására tett törekvések alapvető célja az, hogy olyan lehetőségeket keressünk és találjunk, amelyek mellett a feladat méretei nem jelentik a megoldás akadályát. Teljesen érthető, hogy ilyen cél mellett nem lehet csupán a feladat méretére korlátozni figyelmünket. Szükséges, hogy alaposan elemezzük a feladat szerkezetét is.

A következőkben néhány jellegzetes, a programozási feladatokban gyakran előforduló speciális szerkezet alapján tipizáljuk a nagyméretű feladatokat. A struktúra ebben az összefüggésben a feladat korlátozó feltételeire vonatkozó együtthatómátrix szerkezetére utal. Természetesen a felsoroltakon kívül még számos egyéb típus is lehet, de figyelmünket azokra irányítjuk, amelyek gyakoriak, fontosak és főképp ismert eljárásokkal számítástechnikai célokra felhasználhatók.

A nagyméretű feladatok az együtthatómátrix speciális szerkezete szempontjából két nagy csoportba sorolhatók. Egyik csoportba tartoznak azok a feladatok, amelyek speciális szerkezettel rendelkeznek, a másik csoportba azok, amelyek nem rendelkeznek ilyen szerkezettel. A teljesség igénye nélkül néhány speciális típust ismertetünk. Ezek a következők:

— dinamikus szerkezet,
— többszektoros szerkezet,
— blokkháromszögű szerkezet,
— határolt blokkdiagonális szerkezet.

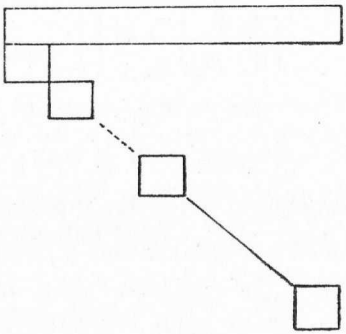
A dinamikus szerkezet a többperiódusú optimalizálási feladatok együtthatómátrixának jellegzetes formája. Az egyes évekre vonatkozó változók és feltételek elrendezése alapján lépcsős szerkezetnek is szokták nevezni. A szemléltetés megkönnyítésére a következő ábrát adjuk:



1. *ábra*

Ilyen típusú modell pl. [6]-ban található. A többszektoros szerkezetet másképpen blokkdiagonálisnak is nevezik. Ábrája a következő:

(1.2)



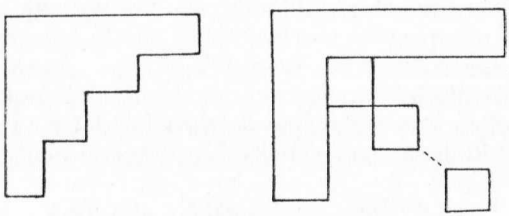
2. *ábra*

A többszektoros elnevezés onnan adódik, hogy a diagonálishoz tartozó blokkokat tekinthetjük úgy, mint egy gazdaság szektorait reprezentáló blokkokat, amelyek a feltételek egy részén keresztül egymással összekapcsolódnak (központi blokk).

A blokkdiagonális szerkezet egy speciális esete lehet az az eset, amelynél az összekapcsolódó, ún. interszektorális kapcsolatok hiányoznak.

A blokkháromszögű és határolt blokkdiagonális szerkezetet ábrával szemléltetjük:

(1.3)



3. *ábra*

Mint említettük a fenti szerkezettípusok a feladatok megoldása szempontjából fontos specialitások. A feladatok nagy csoportja nem rendelkezik ilyen szerkezettel. Előfordul, hogy az adott feladatot át tudjuk alakítani valamelyik speciális szerkezet mintájára, s ezzel valamelyik speciális szerkezetre kidolgozott algoritmust alkalmazni tudunk.

A lineáris programozási feladatok megoldásában alapvető jelentőségű blokkdiagonális szerkezetet például könnyen elérhetjük a következő egyszerű megfontolással. Adott a következő lineáris programozási feladat:

$$x \geqq o$$

$$\begin{pmatrix} A_1 \\ A_2 \end{pmatrix} x \leq \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \tag{1.4}$$

$$f = cx \to \max_x$$

A feladatot a következő formában is megadhatjuk:

$$x_1, x_2 \geqq o$$

$$\begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leqq \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \tag{1.5}$$

$$Ex_1 - Ex_2 = o$$

$$cx_1 + cx_2 = 2f \to \max_x$$

Ugyanez az ötlet alkalmazható a feladat duálisára. A feladat együtthatómátrixa így blokkdiagonálissá vált. A fenti felbontást didadikus felbontásnak nevezzük [23].

Az alkalmazás szempontjából szükséges lenne olyan eljárásokat ismerni, amelyek a feladat speciális struktúráját felderítik. Ebben a kérdésben, valamint az ilyen struktúrák előállítására vonatkozó eljárások tekintetében a gyakorlati készségünkre vagyunk utalva, mivel ilyen algoritmusok nem állnak rendelkezésre.

2. Megoldási módszerek

A nagyméretű feladatok megoldására tett kísérletek eredményeit tárgyaló irodalom rendkívül gazdag. A javasolt megoldási eljárások két részre tagolhatók:

1. Olyan eljárásokra, amelyek az ismert és hatékony eljárásnak nevezhető szimplex módszer további finomítására irányulnak és céljuk a nagyméretű feladatok speciális szerkezetéből adódó lehetőségek felhasználása a számítástechnikai hatékonyság fokozására. Az e csoportba tartozó eljárásokat a továbbiakban *bázismanipulációs* eljárásoknak nevezzük.
2. Olyan eljárásokra, amelyek célja az adott feladathoz tartozó új megoldási algoritmusok kidolgozása. Ilyenek pl. a *dekompozíciós algoritmusok*.

2.1. *Bázismanipulációs eljárások*

Az e csoportba tartozó eljárások lényege olyan lehetőségek keresése, amelyek felhasználásával a szimplex eljárás iterációi kisebb időigénnyel és ugyanakkor kisebb memória-igénybevétellel elvégezhetők. Ennek a célnak az elérése azáltal válik lehetségessé, hogy a feladatok nagy többsége sajátságos szerkezettel rendelkezik. Az egyik adottság, amit itt kihasználhatunk az, hogy a nagyobb feladatok együtthatómátrixának telítettsége alacsony. Nem szerencsés ezzel kapcsolatban számszerű megállapítást tenni, de szemléltetésre Dantzigot idézzük. „Például egy 200 egyenletet tartalmazó feladatnál az 5%-os telítettség a tipikus; nagyobb feladatoknál a telítettség 0,5%-ra vagy az alá esik." [6]. A másik adottság pedig a feladatoknak az 1. pontban említett szerkezettípusokkal való rokonsága.

Mivel a bázismanipulációs eljárások száma rendkívül nagy, arra teszünk kísérletet, hogy csoportosítást készítsünk, majd az eljárások felhasználhatóságának lehetőségeire hívjuk fel a figyelmet. A problémakör áttekintésére szolgáló tanulmányok Balinski [3], Gomory [12] munkái (magyar nyelven: Prékopa-Majthay [21]).

A bázismanipulációs eljárásoknak két csoportja van:
(1) oszlopgenerálási eljárások,
(2) bázisdekompozíciós módszerek.

2.1.1. *Oszlopgeneráló eljárások*

A szimplex módszernél minden egyes iterációban dönteni kell arról, melyik nembázis változót vonjuk be a bázisba. Ez megköveteli az ún. „redukált költség" tényezők kiszámítását és azok vizsgálatát. Amennyiben a változók száma nagy, e vizsgálat elvégzése nagy idő- és memóriafelhasználást igényel. Ennek kiküszöbölése céljából született meg a szimplex algoritmus olyan módosítása, amelynél a direkt számítás helyett a feladat speciális szerkezetéhez igazodó algoritmus szabályozza a bázisba kerülő oszlopok sorrendjét. Innen az oszlopgenerálás elnevezés. Az e területen született első eredmény Ford és Fulkerson nevéhez fűződik, akik 1958-ban publikálták módszerüket [10]. Bizonyos közelítésben a Dantzig—Wolfe-eljárás is úgy interpretálható, mint oszlopgeneráló módszer, hiszen itt is a bázisba kerülő új változók „sorsáról" döntünk az alfeladatok felhasználásával. (Egyéb felhasználásukról l. pl. [10], [12].)

2.1.2. *Bázisdekompozíciós eljárások*

Az 1. pontban vázolt szerkezettípusok figyelembevételével a programozási feladatok szimplex módszerrel való megoldását úgy is megszervezhetjük, hogy mind a számolási időigényt, mind a kapacitásigénybevételt csökkentjük. Az aktuális bázis szerkezetének kihasználására irányuló fogásokat együttesen *bázisdekompozíciónak* nevezzük.

A szimplex algoritmus alkalmazása esetén az aktuális bázis fontos szerepet játszik. Jelöljük B-vel az aktuális bázisvektorokból képzett matrixot. Ennek felhasználásával a következő vizsgálatokat kell elvégezni:

— meg kell határozni a bázismegoldást, azaz meg kell oldani a

$$Bx_B = b$$

egyenletrendszert, ahol x_B a bázismegoldás,

— meg kell határozni a bázisba épő vektornak a pillanatnyi bázisra vonatkozó koordinátáit, azaz meg kell oldani a

$$Bd_k = a_k$$

egyenletrendszert, ahol a_k a belépő vektor,

— meg kell határozni a bevonandó változót.

A fenti munkáknál az aktuális bázis inverzét kell megadnunk. A bázisinverz memórialekötése igen nagy lehet, mivel a ritka matrixok inverze általában a 100%-os tömöttséghez vezet. Nézzünk egy egyszerű példát! Legyen a bázis adott

$$B = \begin{pmatrix} 4 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

Ennek inverze:

$$B^{-1} = \begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & 2 & 1 & 1 \\ -1 & 1 & 2 & 1 \\ -1 & 1 & 1 & 2 \end{pmatrix}$$

Az ilyen típusú mátrixok sajátságos szerkezetét feltétlenül ki kell használni az inverz meghatározásánál. E törekvések eredménye az inverz szorzatformás tárolása, az ún. *szorzatformás inverz*. A szorzatformában való tárolás azt jelenti, hogy az inverz mátrixok elemi mátrixok szorzataként ábrázoljuk. Az elemi mátrix olyan mátrix, amely csak egy sorában vagy oszlopában különbözik az egységmátrixtól. A fenti példára alkalmazva:

$$B^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & -1 & -1 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A számítógépben elegendő csupán az egységmátrixtól eltérő vektorokat tárolni. Természetesen sok iteráció után az elemi mátrixok száma nagy lehet és ez még stabilitási problémákkal is párosulhat, azonban ezek elkerülésére alkalmas fogások állnak rendelkezésre. A szorzatformás inverz alkalmazásának alapos tárgyalása Dantzig könyvében található [8].

A tulajdonképpeni bázisdekompozíció fogalmának azok az eljárások felelnek meg, amelyeknél a bázist bontjuk fel kisebb részekre. A bázis speciális szerkezete ugyanis megengedi, hogy kisebb részének és annak inverzének, valamint pótlólagos információknak a felhasználásával tulajdonképpen a teljes inverzet meghatározhassuk. A bázisnak ezt a részét munkabázisnak („working basis") nevezzük, amely rendelkezik azzal az előnyös tulajdonsággal, hogy egyszerűbben kezelhető mint a tényleges bázis. A munkabázissal kapcsolatos átfogó tanulmány kidolgozása Dantzig nevéhez fűződik [8].

A bázisdekompozíció jól alkalmazható:

— dinamikus struktúrára. Dantzig 1955-ben publikált cikkében már foglalkozik ezzel a kérdéssel [9], majd 1963-ban írott cikke újabb módszert ad arra vonatkozólag, hogyan lehet az inverzet meghatározni és az egyik iterációról a másikra való áttérésnél ezt a tulajdonságot megtartani [8]. Az ilyen struktúrára vezetnek a dinamikus Leontief-modell és a Markov folyamatok speciális esetei,

— blokkdiagonális szerkezetre [6], [8],

— határolt blokkdiagonális szerkezetre [6].

A bázisdekompozíció megvalósítása során a következő körülményeket kell figyelembe venni:

a) Fel kell ismerni a bázis speciális szerkezetét.

b) A szimplex iterációhoz szükséges feladatokat el kell végezni.

c) Az iterációval biztosítani kell, hogy a bázis szerkezete ne változzék.

A bázisdekompozíciós eljárások legegyszerűbb alkalmazási területei az ún. felsőkorlátos technikák [9], [25]. A felsőkorlát technikák alkalmazásának tapasztalatairól beszámoló cikkek szerint e manipulációval a számítási időt esetenként lényegesen csökkenteni lehet.

2.2. *Dekompozíciós algoritmusok*

A dekompozíciós algoritmusok definícióját széles értelemben használjuk. Nem csupán az eredeti feladat felbontás útján való megoldási módszereit soroljuk a módszerek e csoportjába, hanem az azokkal rokon eljárásokat is. Ez lehetővé teszi, hogy a megoldási lehetőségek tárgyalásában kitérjünk mind a lineáris, mind a nemlineáris programozási feladatokra. Azt azonban kikötjük, hogy csak véges számú változót és feltételt tartalmazó feladatokat vizsgálunk. Az újabb kutatások kitérnek a nem véges számú változós feladatok kezelésére is [4].

A következőkben először általános keretben szemléltetjük a dekompozíciós elv lényegét, majd az ennek alapján kialakított optimalizálási módszerek *alapelemeit* tekintjük át. Az alapelemek feldolgozásával az a célunk, hogy a nagyszámú eljárás lényeges jegyeinek kiemelésével rendszerezési szempontokat alakítsunk ki.

Tekintsük a következő feladatot:

(a)
$$x \geq o$$
$$Ax = b$$
$$Bx = d$$
$$cx \to \min$$

ahol x n elemű vektor,
A $m \times n$ méretű mátrix,
B $l \times n$ méretű mátrix,
c, b és d megfelelő méretű vektorok.

Definiáljuk az n dimenziós térnek egy zárt, konvex, poliedrikus halmazát a következőképp: $K = \{x | Bx = d;\ x \geq o\}$. Mivel $x \geq o$, ezért $K = C + H$, ahol C poliedrikus konus, amelyet véges számú $c^1, c^2, \ldots c^r$ extremális irány generál, azaz

$$C = \{x \mid \sum_{j=1}^{r} \mu_j c^j\ ;\ \mu_j \geq 0\}.$$

Hasonlóképpen H az $e^1, e^2, \ldots e^h$ véges számú extremális pont konvex burka, azaz

$$H = \{x \mid x = \sum_{i=1}^{h} \lambda_i e^i\ ;\ \lambda_i \geq 0\ ;\ \sum_{i=1}^{h} \lambda_i = 1\}$$

A fenti jelölések felhasználásával transzformáljuk az (a) feladatot:

$$\lambda_i, \mu_j \geq 0 \qquad i = 1, 2, \ldots, h \quad j = 1, 2, \ldots, r$$

(b)
$$\sum_{i=1}^{h} \lambda_i p^i + \sum_{i=1}^{r} \mu_j r^j = b$$
$$\sum_i \lambda_i = 1$$
$$\sum_{i=1}^{h} \lambda_i \alpha_i + \sum_j \mu_j \beta_j \to \min_{\lambda, \mu}$$

ahol

$$\alpha_i = ce^i; \qquad \beta_j = cc^j$$
$$p^i = Ae^i; \qquad r^j = Ac^j$$

A transzformált feladatot master feladatnak nevezzük. Mivel a h és r általában nagy, a feladat nagyon sok változót tartalmaz. A dekompozíciós elv kialakításánál abból indulunk ki, hogy e nagyszámú csúcspontból és irányból csak annyit vegyünk figyelembe, amennyire szükségünk van.

Tegyük föl, hogy ismerjük az $I^t \subset I = \{1, 2, \ldots h\}$ és $J^t \subset J = \{1, 2, \ldots, r\}$ indexhalmazokat, amelyekre adott a következő feladat:

$$\lambda_i \geq 0 \qquad i \in I^t$$
$$\mu_j \geq 0 \qquad j \in J^t$$
(c)
$$\sum_{i \in I^t} \lambda_i p^i + \sum_{j \in Jt} \mu_j r^j = b$$
$$\sum_{i \in I^t} \lambda_i$$
$$\sum_{i \in I^t} \lambda_i \alpha_i + \sum_{j \in Jt} \mu_j \beta_j \to \min$$

Ezt a feladatot redukált master feladatnak fogjuk nevezni, mivel formailag a master feladathoz hasonlít, de oszlopainak csak egy részét tartalmazza. A redukált master feladatról feltesszük, hogy rendelkezik lehetséges bázismegoldással (azaz, $m + 1$ lineárisan független oszlopa van, valamint a megfelelő λ_i és μ_j értékek nem-negatívok). Feladatunk most már az, hogy e redukált master feladat megoldásának felhasználásával eljussunk az eredeti feladat megoldásához. Ennek érdekében a következő megfontolásokat tesszük:

Jelöljük $I^{t,0} \subset I^t$ és $J^{t,0} \subset J^t$ szimbólumokkal a redukált feladat optimális bázisindexeit, (π^t, δ_t)-vel a duál megoldásokat. Ekkor

$$\pi^t p^i + \delta_t = \alpha_i \qquad i \in I^t,$$
$$\pi^t r^j = \beta_i \qquad j \in J^t,$$

A (π^t, δ_t) szimplex együtthatókat felhasználjuk az egész feladat optimalitásának vizsgálatára. Keresünk olyan p^i-ket $(i \in I - I^t)$, amelyekre $\pi^t p^i + \delta_t > \alpha_i$ vagy olyan r^j-ket $(j \in J - J^t)$, amelyekre $\pi^t r^j > \beta_j$. A K definíciója alapján tehát olyan e^i-ket keresünk, amelyekre $(\pi^t A)\, e^i + \delta_t > ce^i$ vagy c^j-ket, ahol $(\pi^t A)\, c^j > cc^j$. Ehhez a következő kisebb méretű ún. alfeladatot oldjuk meg:

$$x \geq o$$
$$Bx = d$$
(d)
$$(c - \pi^t A)x + \delta_t \to \min_x$$

A szimplex algoritmus ismert tulajdonsága alapján tudjuk, hogy a (d) megoldása során az alábbi esetek fordulnak elő:

1. A (d)-nek van optimális megoldása. Az optimális megoldás e^t lesz. Ebben az esetben kiszámítjuk az Ae^t vektort, amellyel mint új oszloppal a (c) feladatot bővítjük és azt ismételten megoldjuk.

2. A (d) célfüggvénye alulról nem korlátos $x \in K$-ra. Ismeretes, hogy ekkor létezik olyan $c^t \geq o$, amelyre $Bc^t = o$ és $[c - (\pi A)]\, c^t < o$. Ebben az esetben az $r^t = Ac^t$ oszloppal bővítjük a (c) feladatot és megoldjuk.

3. A (d) feladatnak nincs lehetséges megoldása. Ebben az esetben az eredeti feladatnak sincs lehetséges megoldása.

A fenti gondolatmenet szerint az (a) megoldáshoz egy iteratív folyamat segítségével jutunk. Az iteratív folyamatban a redukált master feladat és az alfeladat közötti információáramlásnak van döntő szerepe. Az alfeladattal elérhetjük, hogy az extremális csúcsok és irányok közül csak egyeseknek a meghatározását kell elvégeznünk, míg eljutunk az (a) megoldásához.

Az eljárás különösen akkor hatékony, ha az A-nak kevés sora van, az B pedig speciális szerkezetű. Ilyen egyszerű eset az, amikor az B blokkdiagonális szerkezetű.

A gondolatmenet kiterjeszthető olyan esetekre is, amelyeknél a K extremális pontjainak száma nem véges. Ezekkel itt nem foglalkozunk.

Az irodalomban fellelhető módszerek nagy részében felismerhetők a fenti, általánosabban vázolt megoldási elemek. A következőkben szeretnénk a módszerek legfontosabb elemeit összefoglalni. Ehhez a módszerekben rejlő fő közös tulajdonságok ismeretéből kell kiindulni.

A dekompozíciós algoritmusok legfontosabb közös jellemzői a következők:

— az adott feladat alkalmas átalakítással könnyebben kezelhető formába írható és ezáltal kisebb feladatok megoldására vezethető vissza. A felbontás eredményeképp kapjuk az ún. master feladatot, és az ún. alfeladatokat. Ez a felbontás a tulajdonképpeni dekompozíció,

— az átalakított feladathoz kereshető olyan megoldási fogás vagy stratégia, amely az alfeladatok és a master feladat közötti kapcsolatot teremti meg,

— az alfeladatok és a master feladat között meghatározott információáramlás van,

— az algoritmusok legtöbbször iterációk sorozatából állnak.

A fenti jellemzők kezünkbe adják a dekompozíciós eljárások csoportosítási lehetőségének kulcsát. Az algoritmusok ugyanis jellemezhetők egyrészt az átalakítási fogásokkal (gyakran dekompozíciós szabálynak nevezik), amelyeket feladatorientált manipulációknak fogunk nevezni. Valamely manipuláció egy megoldási stratégiával párosítva egy-egy algoritmust reprezentál. Az irodalomban ismert eljárások legtöbbje négy manipulációt alkalmaz. Ezeket tekintjük át a következőkben, majd utána kitérünk a megoldási stratégiákra. A manipulációk célja

— a feladat felbontása kisebb feladatokra (felbontás),

— a feladat átalakítása olyan szerkezetűvé, amelyre könnyen kezelhető algoritmus van (például vegyes nem lineáris feladatoknál a nem lineáris rész kiküszöbölése) (izolálás),

— a nem lineáris feltételek lineáris közelítése (közelítés).

2.2.1. *Dualizálás*

A legegyszerűbb fogás, amellyel valamely feladat egyszerűbb alakra bontható. A szimplex algoritmus alkalmazásánál akkor használjuk, ha a sorok száma az oszlopok számához viszonyítva nagy. A dekompozíciós módszerek irodalmában Rosen-féle particiós módszer példázza ezt a manipulációt, ahol

a Dantzig—Wolfe-eljárásnál megismert blokkdiagonális szerkezet duálisából indulunk ki [22].

A dualizálásnál a fő problémát a nem lineáris esetek kezelhetősége jelenti. Az elmúlt években több cikk foglalkozik a nem lineáris problémák duálisának kérdéseivel [11].

2.2.2. *Belső linearizálás*

Gyakorta használt manipuláció, amely elsősorban a nem lineáris programozásban lehet eredményes. Valójában a Dantzig—Wolfe-eljárás alapgondolata ennek egy speciális esete, ezért részletesebben foglalkozunk vele.

Tekintsük a következő feladatot:

$$g_i(x) \leq o \qquad i = 1, 2, \ldots, m \tag{2.1}$$

$$f(x) \to \max$$

ahol g_i és f folytonos függvények. Az egyszerűség kedvéért feltesszük, hogy

$$S = \bigcap_i \{x : g_i(x) \leq o\}$$

konvexhalmaz. (Ha minden g_i kvázikonvex, akkor S konvex.)

A 2.1 feladatot másképpen is felírjuk:

$$G(x) \leq o \tag{2.2}$$

$$f(x) \to \max$$

A (2.1) feladat egy nem lineáris programozási feladat. Speciális esetben a g_i függvények lineárisak:

$$g_i(x) = \sum_j a_{ij} x_j - b_i \qquad i = 1, 2, \ldots, m \tag{2.3}$$

azaz

$$G(x) = Ax - b$$

A továbbiakban megfontolásainkat az általánosabb esetre tesszük.

Legyenek $x_1, x_2, \ldots x_T$ n-elemű vektorok. Ezen vektorok konvex burkának valamely x pontja így

$$x = \sum_{t=1}^{T} \alpha_t x_t \tag{2.4}$$

ahol $\sum_t \alpha_t = 1$ és $\alpha_t \geq o \ \forall$ t esetre.

Valamely $g(x)$ függvény e pontok konvex burkán a következő közelítéssel lineraizálható

$$g(x) = \sum_{t=1}^{T} \alpha_t g(x_t) \tag{2.5}$$

Ha a fenti problémában x és annak minden függvénye helyett a fenti (2.4) és (2.5) reprezentációt használjuk, akkor a (2.1) feladat a következő alakú lesz:

$$\sum_t \alpha_t g_i(x_t) \leq 0 \qquad (i = 1, \ldots m) \tag{2.6}$$

$$\sum_t \alpha_t = 1$$

$$\alpha_t \geq 0 \ (\forall t)$$

$$\sum_t \alpha_t f(x_t) \to \max$$

A (2.6) α_t változókkal adott lineáris programozási feladat. Legyenek $\hat{\alpha}_1, \hat{\alpha}_2, \ldots$ $\ldots, \hat{\alpha}_T$ értékek a megoldásai. Ekkor

$$\hat{x} = \sum_t \hat{\alpha}_t x_t \tag{2.7}$$

az eredeti (2.1) feladat közelítő megoldása. Az, hogy ez a közelítés mennyire jó függ egyrészt az x_t pontoktól, de függ az f és g_i függvények természetétől, valamint a (2.6) feladat kezelési módjától.

Ha a g_i függvények konvexek, akkor a konvexitás miatt

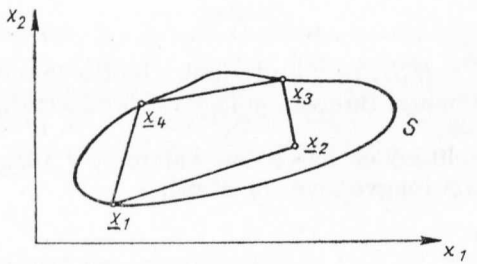
$$g_i(\hat{x}) = g_i(\sum_t \hat{\alpha}_t x_t) \leq \sum_t \hat{\alpha}_t g_i(x_t) \leq 0 \tag{2.8}$$

azaz $\hat{x} \in S$.

Ha az f konkáv, akkor

$$\sum_t \alpha_t f(x_t) \leq f(\hat{x})$$

azaz a (2.6) feladat célfüggvénye az $\hat{x}$-hez tartozó érték alsó korlátja. Világítsuk meg egy példával a R^2-ben a fenti gondolatmenetet. Tekintsük a következő ábrát (1. ábra):



1. *ábra*

Az S halmazt az x_1, x_2, x_3, x_4 pontok konvex burkával közelítjük. A belső linearizálás azt jelenti, hogy az S-et közelítő halmaz konvex poliéder, amelyet véges számú lineáris egyenlőtlenség ír le. Az x_1, x_2, x_3, x_4 pontokat bázisnak nevezzük.

A konvex halmaz mintájára a konvex (konkáv) függvény is linearizálható!

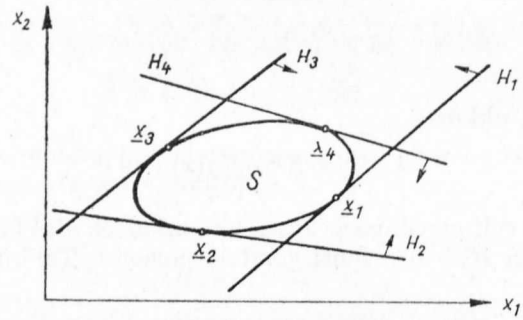
A linearizálás jelentőségét növeli az a körülmény, hogy rugalmasan alkalmazható mind a bázis megválasztását, mind a linearizálandó halmazt illetően.

A fenti kétdimenziós ábra viszonylag könnyűnek tünteti fel a dolgot, és ez kevés változó esetén talán megfelel a valóságnak, de sok változó esetén komplikáltabb. Nagy előnye abban áll, hogy a manipulációt csak implicite kell elvégezni, azaz a bázis pontjainak csak egy kis részét kell meghatároznunk. Valójában a Dantzig—Wolfe-eljárás mögött húzódó gondolat ilyen természetű, ami

lehetőséget ad a DW-eljárás nem lineáris verziójának kifejlesztésére. Abban az esetben, ha maga az S halmaz konvex poliéder, akkor a belső linearizálás elegánsan elvégezhető. Csupán az a teendő, hogy a szóban forgó bázis elemeit a poliéder extremális pontjaival tesszük egyenlővé. Az extremális pontokat csupán a kívánt mértékben kell meghatározni.

2.2.3. *Külső linearizálás*

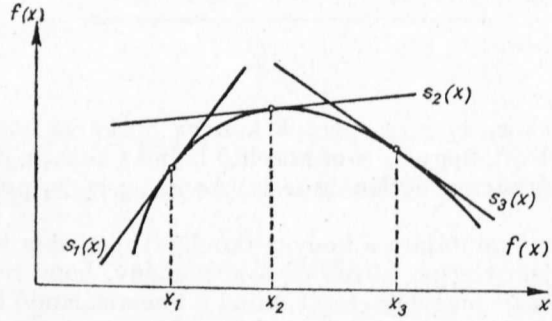
A belső linearizálás azon az alapgondolaton épült föl, hogy az S halmazt bizonyos pontjai konvex burkaként ábrázoljuk. A külső linearizálás pedig úgy tekinthető, mint egy olyan közelítő eljárás, amely az S halmazt félterek halmazának közös részeként ábrázolja. Tartsuk meg továbbra is az S-re adott definíciónkat. Szemléltetésre tekintsük az R^2-ben adott konvex halmaz külső linearizálását (2. ábra):



2. *ábra*

Az S halmazt tehát a $H_1 \ldots H_4$ hipersíkok alkotta félterek közös része által definiált konvex halmazzal (konvex poliéder) közelítettük meg, ahol is a poliéder tartalmazza S-et.

Az S halmaz közelítéséhez hasonlóan valamely f függvény is közelíthető szakaszonként lineáris függvénnyel (3. ábra).



3. *ábra*

Az s függvényeket úgy definiáljuk, hogy adott x_i-nél rendelkezzenek azzal a tulajdonsággal, hogy értékük sehol sem kisebb az f-nél (ha f konkáv) és x_i pontban $s(x_i) = f(x_i)$. Ha az f függvény differenciálható, akkor egy $\hat{x}$-hez

tartozó s függvény az elsőrendű Taylor-sorral adható meg, azaz

$$s(\hat{x}) = f(\hat{x}) + \nabla f(\hat{x}) \cdot (x - \hat{x})$$

ahol $\nabla f(\hat{x})$ az $f(x)$ $\hat{x}$-beli gradiense, $\nabla f(x) = \left(\frac{\vartheta f(x)}{\vartheta x_1}, \ldots, \frac{\vartheta f(x)}{\vartheta x_n} \right)$

Az elmondottakat alkalmazzuk most programozási feladatra! Tekintsük az eredeti (2.1) feladatot és tegyük fel, hogy adottak az $x_1, x_2, \ldots x_T$ pontok, amelyekhez i_t indexet rendelünk 1 és m között. Minden t-re a $g_{i_t}(x) \geq o$ feltételt a Taylor-sorral közelítsük, s ekkor a nem lineáris feladat a következő feladatba megy át:

$$g_{i_t}(x_t) + \nabla g_{i_t}(x_t)(x - x_t) \leq 0 \qquad t = 1, \ldots T \tag{2.7}$$

$$f(x) \to \max_x$$

Feltesszük, hogy minden g_i konvex. Ebben az esetben minden S halmazbeli pont kielégíti a (2.7) feltételt is, mivel a (2.7) baloldala sohasem nagyobb $g_{i_t}(x)$-nél. A (2.7) által definiált halmaz nagyobb S-nél.

A feladat megoldására gondolva látható, hogy a megoldás akkor lehet hatékony, amennyiben a (2.7) az eredeti feladat megoldását anélkül adja, hogy sok x_t meghatározása válna szükségessé. A megoldás egy hatékony módszerét Kelley dolgozta ki [14].

A fenti manipuláció hatékonysága a közelítő pontok gazdaságos generálásától függ. A belső linearizálás alapgondolatához hasonlóan Kelley módszere olyan, amelyben nem kell előre megadni explicite az x_t pontokat, csupán a kívánt mértékben.

A külső linearizálás felvet egy természetesen adódó kérdést: vajon létezik-e adott ponthoz tartozó „érintő". Az már bizonyított, hogy az R^n-ben adott zárt konvex halmaz valamely határpontja legalább egy rajta áthaladó támasz-síkkal rendelkezik [11]. Ebből következik, hogy minden zárt konvex halmaz a határoló félterek közös részeként reprezentálható.

2.2.4. *Vetítés*

Tekintsük a következő egyszerű feladatot:

$$\begin{gathered} y \in Y \\ x - g(y) = 0 \\ h(x) \geq 0 \\ \max_{x,y} f(x, y) \end{gathered} \tag{2.8}$$

ahol a h és g vektorfüggvények. A feladat y változóra való vetítésén a következő feladatot értjük:

$$\max_{y \in Y} \left\{ \begin{matrix} \sup_x f(x, y) \\ h(x) \geq 0, x = g(y) \end{matrix} \right\} \tag{2.9}$$

A kapcsos zárójelben levő supremumot $v(y)$-nal jelölve

$$v(y) = \begin{cases} f[g(y), y)] & \text{ha} \quad h[g(y)] \geq 0 \\ -\infty & \text{ha} \quad h[g(y)] = 0 \end{cases} \tag{2.10}$$

Bevezetve a $V = \{y | h[g(y)] \geq 0\}$ jelölést, a (2.8) feladat a következőképp írható

$$\begin{gathered} y \in V \cap Y \\ v(y) \to \max. \end{gathered} \tag{2.11}$$

A fenti példa szemlélteti a *vetítés* vagy másképpen a *particionálás* lényegét. Látható, hogy a vetítéssel az eredetinél egyszerűbb feladathoz jutottunk azáltal, hogy a változók egy részét (az y változókat) átmenetileg rögzítettük.

A módszer általánosítását most már könnyen elvégezhetjük. Tekintsük most az alábbi feladatot:

$$\begin{gathered} x \in X \\ y \in Y \\ g(x, y) \geq 0 \\ \max f(x, y) \end{gathered} \tag{2.12}$$

ahol $X \subseteq E^n$, $Y \subseteq E^m$ és g egy vektorfüggvény. A (2.12)-nek az y változók terére eső vetületét a következők szerint definiáljuk:

$$\max_{y \in Y} \begin{Bmatrix} g(x, y) \geq 0 \\ \sup\limits_{x} f(x, y) \end{Bmatrix} \tag{2.13}$$

A (2.13) már csak y-ban változó feladat. Ha a kapcsos zárójelben levő kifejezést $v(y)$-nal jelöljük, akkor a feladatot átírhatjuk. Ha valamely y esetén $v(y)$ nem megengedett, akkor v-t $-\infty$ értékűnek tekintjük. Legyen $V = \{y | g(x, y) \geq 0,\ x \in X\}$. A (2.13) feladat tehát

$$\begin{gathered} y \in V \\ \max_{y \in V} v(y) \end{gathered} \tag{2.14}$$

A V halmazt az $x \in X$ és $g(x, y) \geq 0$ feltételek y terére eső vetületének nevezzük.

A (2.12) és (2.14) közötti nagyon fontos összefüggést a következő tétel írja le:

TÉTEL: A (2.12) akkor és csak akkor nem megengedett vagy nem-korlátos, ha a (2.14) is az. Ha $(y_0 y_0)$ a (2.12) optimális megoldása, akkor y_0 a (2.14) optimális megoldása. Ha y_0 optimális megoldása a (2.14)-nek és x_0 eléri az $f(x, y_0)$ supremumot az $x \in X$ és $g(x, y_0) \geq 0$ mellett, akkor x_0 az y_0-al együtt optimális megoldása a (2.12)-nek.

A vetítés manipulációját alkalmazhatjuk például a következő feladat esetén:

$$\begin{gathered} x_k \in X_k \qquad k = 1, \ldots, K \\ y \in Y \end{gathered} \tag{2.15}$$

$$G_k(y) + g_k(x_k) \leqq 0 \qquad (k = 1, \ldots, K)$$

$$\max_{x,y} \{F(y) + \sum_{k=1}^{K} f_k(x_k)\}$$

ahol x_k n_k dimenziós vektor és G_k, valamint g_k ugyanazon méretű vektorfüggvények. A feladat egy K szektoros modellként interpretálható, ahol az y változók a központilag ellenőrzött tevékenységeket szimbolizálják, míg az x_k a szektorok változóinak vektora. Ha az y változókat átmenetileg rögzítjük, akkor K független feladatot kapunk, amely az y-ra való vetítésnek felel meg:

$$\max_{y \in Y} [F(y) + \sum_{k=1}^{K} v_k(y)]$$

(2.16) ahol

$$v_k(y) = \left\{ \begin{matrix} \sup f_k(x_k) \\ x_k \in X_k \\ g_k(x_k) \leqq -G_k(y) \end{matrix} \right\}$$

A vetítés további alkalmazását a Benders-féle particiós módszer kapcsán mutatjuk be [2.2.6 pont].

A fent bemutatott átalakítási fogások felhasználási körét a következő tábla szemlélteti:

	Felbontás	Izolálás	Közelítés
1. Dualizálás	+	+	
2. Belső linearizálás	+		
3. Külső linearizálás			+
4. Vetítés		+	

Érdekes annak összefoglalása is, hogy az ismertebb algoritmusok milyen manipulációkat alkalmaznak.

	Dualizálás	Belső linearizálás	Külső linearizálás	Vetítés
Abadie—Williams [1]		+		
Benders [26]			+	+
Dantzig—Wolfe [5]		+		
Rosen [22]	+		+	
Weitzman [24]			+	+
ten Kate [13]		+		

2.2.5. *Megoldási stratégiák*

Áttekintettük a legfontosabb átalakítási fogásokat, amelyekkel elértük, hogy az adott feladatot könnyebben kezelhető alakra hoztuk. Ehhez mindig választható olyan megoldási stratégia, amelynek felhasználásával a feladatokat egyszerűbb optimalizálási feladatok — ún. alfeladatok — megoldásának sorozatával állítjuk elő. Amennyiben a feladat szerkezete valamilyen speciális

szerkezet-típusba tartozik, úgy az alfeladatok kialakítása során ezt a tulajdonságot kihasználjuk.

A megoldási stratégiák száma rendkívül nagy, s ezek nagy része kiválóan alkalmazható a nagyméretű feladatok megoldásában. Tekintsünk át röviden néhány fontosabb megoldási stratégiát:

a) *Relaxációs stratégia*

A relaxáció olyan konvex feladatok esetén célravezető eljárás, amelyek sok egyenlőtlenséget tartalmaznak. A módszer alapgondolata a következőképpen foglalható össze. Megoldjuk az adott feladat egy olyan változatát, amelyet úgy kapunk, hogy az egyenlőtlenségek egy részét (lehetőleg minél többet) nem vesszük figyelembe. Ha az így kapott megoldás nem elégíti ki az elhagyott feltételeket, akkor generálunk és a vizsgálati körünkbe vonunk egy ilyen feltételt és az lgy kapott feladatot oldjuk meg. A bővített feladat megoldását behelyettesítjük az ignorált feltételekbe. Ha a megoldás kielégíti a feltételeket, akkor az eljárást befejezzük, ha viszont nem ez az eset áll fenn, akkor az előbbi bővítést alkalmazva újra megoldjuk a feladatot. A lineáris programozási feladatok megoldásában is ismert Lemke-féle duál módszer a relaxációs stratégiával ekvivalens.

A stratégiát a Kelley által kidolgozott metszősík módszeren szemléltetjük.

A megoldandó feladat:

$$x \geq 0$$
$$Ax \leq b$$
$$g(x) \leq 0$$
$$cx \to \min$$

ahol a g-ről feltesszük, hogy konvex és differenciálható az $X = \{x | x \geq 0, Ax \leq b\}$ halmazon.

A külső linearizálást alkalmazva a feladat a következő feladattá alakítható:

$$x \in X$$
$$g(\tilde{x}) + \nabla g(\tilde{x}) \leq 0 \quad \forall\, \tilde{x} \in X$$
$$cx \to \min$$

A relaxációs stratégia lehetővé teszi, hogy kezdetben kerüljük a g valamennyi érintőjének meghatározását. Minden iterációban e feladat egy csökkentett változatát oldjuk meg. A csökkentett feladat egy x^* optimális megoldása akkor és csak akkor megengedett az átalakított feladatban, ha $g(x^*) \leq 0$. Ha $g(x^*) \not\leq 0$, akkor a $\nabla g(x^*)$ felhasználásával előállítható egy megsértett feltétel, amelyet a csökkentett feladathoz csatolunk. A stratégia alkalmazását megkönnyíti az a körülmény, hogy a csökkentett lineáris programozási feladat módosítása után nem kell a számításokat előlről kezdeni, hanem postoptimalizációs technikával állapíthatjuk meg az új megoldásokat.

b) *Lépcsőzetes közelítés* [11], [22]

A megoldási stratégia nagyméretű feladatok megoldásában először J. B. Rosen particiós módszerénél fordult elő [22]. Általában olyan feladatok megoldásánál előnyös, amelyek lényegesen egyszerűbbé válnak akkor, ha a bennük

szereplő változók egy részét átmenetileg rögzítjük. Ilyen átmeneti rögzítést tudunk elérni az értelmezési tartományok valamely változóra való veítésével. A később bemutatandó Benders-módszer erre is jó példa. Általánosságban is igaz, hogy ez a stratégia a vetítéssel párosítható előnyösen. Eltekintünk a további részletezéstől, a fenti irodalmi utalásokban részletes leírás található.

c) *Restrikciós stratégia* [5], [10], [11]

Olyan feladatok esetén, amelyeknél sok nem-negatív változóval kell dolgoznunk, hatékonyan alkalmazhatjuk. Valójában a lineáris programozás szimplex módszere képviseli ezt a stratégiát. Most egy kicsit általánosabban értelmezzük a lényegét: a változók egy részhalmazát átmenetileg 0-val tesszük egyenlővé és a csökkentett feladatot megoldjuk. Ha a kapott megoldás nem elégíti ki az adott feladat optimumkövetelményét, akkor az előbb megkötött változók közül egyet vagy többet „felszabadítunk" — azaz megengedjük a pozitivitást — és a most már kisebb kötöttséggel bíró feladatot megoldjuk. Az eljárást addig folytatjuk, míg az optimumkritérium teljesül. A megoldási stratégia alapgondolata a Dantzig—Wolfe-eljárásban megtalálható, ahol a master feladat felírása során az extremális pontok nagy száma miatt nagyon sok változós feladatot kapunk. A restrikciós stratégia kiváló tárgyalása található a [11]-ben.

A bemutatott manipulációk, valamint a megoldási stratégiák párosítása megteremti a kb. 60—70 ismertetett eljárás osztályokba sorolásának keretét. Valamely stratégia általában több manipulációval is párosítható. Ez magyarázza a jelenleg meglevő nagyszámú algoritmust. Az osztályozási szemponton túl az új algoritmusok kidolgozásának módját is megkapjuk ezzel a párosítással.

Nem célunk itt valamennyi eljárást osztályokba sorolni. Inkább néhány közismert eljárást jellemzünk a manipulációk és stratégiák párosítása szerint. A Dantzig—Wolfe-eljárás így a belső linearizálás és a restrikciós stratégia párosításának felel meg. Benders módszere a vetítés és fokozatos közelítés párosítását reprezentálja, ugyanígy Weitzman módszere is.

Vannak azonban olyan eljárások, amelyek nem feleltethetők meg valamely párosításnak, mert nem a fenti átalakítási fogásokat alkalmazzák. Ilyen esetben a módszerek jellemzését úgy tehetjük teljessé, hogy az egyéb tulajdonságaikat is megadjuk. A master és az alfeladatok közötti iterációs kapcsolatban áramló információk alapján történő csoportosításról a 3. pontban írunk.

Összefoglalásképpen legszögezhetjük, hogy a dekompozíciós eljárások jellemzését a következő ismérvek alapján végezhetjük:

1. a master feladat kialakításánál használt manipulációk jellege,
2. az alfeladatok megoldásában használt megoldási stratégiák
3. a master és az alfeladatok közötti információáramlás jellege.

2.2.6. *Két példa*

1. *Benders módszere* [26]

Megoldandó a következő feladat:

(2.17) $$x \geqq 0$$
$$y \in Y$$

$$Ax + F(y) \leqq b$$
$$cx + f(y) \to \max_{x,y}$$

A feladat valójában egy szemi-lineáris programozási feladat. Jó lenne ha az y változókat átmenetileg rögzítenénk, mert úgy egyszerűbb feladathoz jutnánk. Alakítsuk át ezért a (2.17) feladatot úgy, hogy az y-ra való vetítést alkalmazzuk. Ekkor

$$\max_{y \in Y} \{f(y) + \sup_{x \geqq 0} [cx, Ax = b - F(y)]\} \tag{2.18}$$

A kapcsos zárójelben levő supremum a következő lineáris programozási feladatnak felel meg:

$$x \geqq 0$$
$$Ax \leqq b - F(y) \tag{2.19}$$
$$cx \to \max$$

A feladat y változásától függően jobboldalban paraméteres feladat. Feltesszük, hogy legalább egy olyan y létezik, ahol véges optimuma van, ekkor a dualitási tétel értelmében igaz továbbá, hogy az

$$u \geqq 0$$
$$uA \geqq c \tag{2.20}$$
$$u[b - F(y)] \to \min$$

megengedett minden y-ra. Jelöljük $\langle u^1 \ldots u^p \rangle$-vel a (2.20) extremális pontjait és $\langle u^{p+1} \ldots u^{p+q} \rangle$-val az extremális irányait. A dualitási tétel értelmében igaz továbbá, hogy a (2.19) akkor és csak akkor megengedett, ha a (2.20)-nak van véges optimuma, azaz, ha y kielégíti az

$$u^j[b - F(y)] \geqq 0 \qquad j = p + 1, \ldots, p + q \tag{2.21}$$

feltételeket. Mivel a (2.18)-ban mindazon y-ok esetén $-\infty$ értéket tekintünk, amelyre (2.19) nem megengedett, ezért a (2.21) feltételeket a (2.18)-hoz tehetjük. A (2.18) helyett a dualitási tételen alapuló megfontolások alapján a következőt írhatjuk:

$$\max_{y \in Y} \{f(y) + \min_{1<j<p} [u^j(b - F(y))]\} \tag{2.22}$$
$$u^j(b - F(y)) \geqq 0 \qquad (j = (p + 1), \ldots (p + q)$$

Mivel a minimum a legnagyobb alsó korlát, a feladat a következő lesz:

$$\underset{y \in Y; y_0}{\text{maximum}} f(y) + y_0$$
$$y_0 \leqq u^j(b - F(y)) \qquad j = 1, \ldots, p \tag{2.23}$$
$$0 \leqq u^j(b - F(y)) \qquad j = p + 1, \ldots, p + q$$

Nézzük most meg a fenti megfontolások alapján kialakított feladat megoldási algoritmusát. Az algoritmus lényege az, hogy a feladatot felbontjuk kisebb

feladatok megoldására, s ezáltal elkerüljük, hogy kezdetben minden u^j-t ismerjünk.

1. lépés: meghatározunk egy $(\hat{y}_0, \hat{y})$ értékpárt, ahol $\hat{y} \in Y$.
2. lépés: megoldjuk a (2.20) feladatot, amelynek eredményeképp kapjuk az u megoldásokat. (A 2.20)-ban y helyére $\hat{y}$ írunk.)
3. lépés: Vizsgáljuk a következő relációt (optimumszabály)

$$y_0 \leqq u^j[b - F(y)]$$

Két eset van:

(a) $\hat{y}_0 \leqq u^j[b - F(y)]$
Ebben az esetben $(\hat{y}_0, \hat{y})$ optimális megoldása az induló feladatnak.

(b) Ha az (a) reláció nem teljesül, akkor előállítjuk a (2.23) egy megsértett feltételét, amellyel bővítjük a feladatot.[1]

2. *Dantzig—Wolfe dekompozíciós eljárás* [5]

A szerzők egy speciális struktúrájú lineáris programozási feladatra alkalmazták a dekompozíció elvét.

A feladat a következő:

$$\begin{aligned} x &\geqq 0 \\ Ax &= a \\ Bx &= b \\ cx &\to \max \end{aligned} \tag{2.24}$$

A $Bx = b$ feltételek tartalmazzák az ún. közös feltételeket, az $Ax = a$ feltételek az ún. speciális feltételeket akkor, ha az együtthatómátrixot blokkdiagonálisnak tételezzük fel. A következő megfontolásokon keresztül megvilágítjuk a dekompozíciós elv lényegét.

Definiáljuk a $P = \{x | Ax = a,\ x \geqq 0\}$ poliédert. A kiinduló feladat ekkor átírható:

$$\begin{aligned} x &\in P \\ Bx &= b \\ cx &\to \max \end{aligned} \tag{2.25}$$

A feladatot átalakítottuk. Föltesszük, hogy a P nem üres és korlátos, s extremális pontjait $x_1, \ldots x_p$ halmaz reprezentálja. A (2.24) a belső linearizálás felhasználásával átfogalmazható

$$\begin{aligned} \alpha &\geqq 0 \\ B \sum_{j=1}^{p} \alpha_j x_j &= b \end{aligned} \tag{2.26}$$

[1] A kifejtésben szándékosan nem írtunk a halmaz természetéről. Szeretném ugyanis kiemelni az eljárásnak azt a jellemzőjét, hogy az Y definiálásától függően a módszer más és más típusú feladatok megoldását teszi lehetővé. Ha az Y egy konvex poliéder integer pontjaiból áll és f, F lineáris, akkor a (2.23) integer lineáris feladat. Ha rendelkezünk ilyen feladat megoldására alkalmas gépi algoritmussal, akkor a Benders-módszer hatékonyan alkalmazható.

$$\sum_{j=1}^{p} \alpha_j = 1$$

$$c(\sum_{j=1}^{p} \alpha_j x_j) \to \max$$

A (2.24) feladat ekvivalens a (2.26)-tal. A feladat felírásához ismernünk kellene valamennyi extremális megoldást. A belső linearizálás gondolatát felhasználva elérhetjük, hogy nem kell valamennyi extremális pontot explicite megadni. Erre szolgál egy olyan iterációs eljárás, amelyben a (2.26) tölti be a master feladat funkcióját, míg az alfeladathoz a következő megfontolások útján jutunk. Jelöljük a (2.26) feladat duál változóit (u, u_0)-val, ahol u_0 a normalizáló feltételhez tartozó duál változó. A programozási feladat optimalitási kritériuma szerint a redukált költségnek nem negatívnak kell lenni, azaz

$$u_0 + uBx^j - cx^j \geq 0 \qquad (j = 1, 2, \ldots p) \tag{2.27}$$

amely ekvivalens az

$$u_0 + \min_{1 \geq j \geq p} (uB - c)\, x^j \geq 0 \tag{2.28}$$

vagy, mivel egy lineáris függvény a P halmazon csúcspontban veszi fel a minimumát,

$$u_0 + \min_{x \in P} (uB - c)\, x \geq 0 \tag{2.29}$$

relációval.

A (2.29)-ből az

$$\begin{gathered} x \geq 0 \\ Ax = b \\ \min\, (uB - c)\, x \end{gathered} \tag{2.30}$$

feladatot nevezzük alfeladatnak, amelynek alapvető szerepe van az algoritmusban.

Az algoritmus így a következő:

1. lépés: meghatározzuk a (2.26) egy lehetséges bázismegoldását és a kapcsolódó (u, u_0) értékeket.
2. lépés: Megoldjuk a (2.30) feladatot.
3. lépés: Vizsgáljuk a következő relációt

$$u_0 + \min_{x \in P} (uB - c)\, x \geq 0$$

Két eset van:

a) az egyenlőtlenség teljesül, ekkor elértük a kiinduló feladat optimumát

b) nem teljesül az egyenlőtlenség, ekkor bővítjük a feladatot a következő oszloppal:

$$\begin{pmatrix} Ax^{j_0} \\ 1 \\ cx^{j_0} \end{pmatrix}$$

Az iterációt a 4. lépésnél folytatjuk.

4. lépés: A bővített feladatot megoldjuk és a 2. lépésnél folytatjuk az iterációt.

3. A dekompozíciós eljárások felhasználása

Ha összefoglaljuk a dekompozíciós módszerek fő jellemzőit, akkor a módszerek érdekes közgazdasági interpretációjához juthatunk. A nagyméretű feladatot alfeladatokra bontottuk, valamint egy ún. master feladatra. A megoldási algoritmus egy iterativ folyamat a master és alfeladatok között, amelyben meghatározott információk áramlanak az egyes feladattípusok között. Az iterációs folyamatnak akkor van vége, amikor elértük a kiinduló feladat globális optimumát. A dekompozíciós módszereknek ezt a leírását rokonságba hozhatjuk a tervezési folyamatok egyszerűsített leírásával. A gazdaságról feltételezzük, hogy egy központra és több szektorra tagozódik. A tervezési folyamat célja a gazdaság egészére vonatkozó optimalizálási feladat megoldása. A központ nem képes egyedül megoldani a teljes feladatot. A számításokat megszervezheti úgy, hogy a nagy feladatot felbontja (dekomponálja) kisebb feladatokra, méghozzá a gazdaság szervezeti tagozódásának megfelelő szektorfeladatokra. A tervezési folyamatban a központ problémája ekkor az, hogy úgy szervezze meg a szektorok és központ közötti információcsere folyamatát, hogy a szektorfeladatok optimális megoldásai az egész gazdaságra vonatkozó optimumhoz konvergáljanak.

A fenti rokonság szemléltetésére tekintsük a következő feladatot:

$$\sum_{i=1}^{K} g_i^j(x_i) \leq b_0^j \qquad (j = 1, \ldots m) \tag{3.1}$$

$$\sum_{i=1}^{K} c_i(x_i) \to \max$$

ahol az x_i az egyes szektorok n_k méretű vektorai, K a szektorok száma.

Tételezzük fel, hogy a tervezési folyamat kezdetén a szektorok csak a saját $g_i^j(x_i)$ és $c_i(x_i)$ függvényeit ismerik, de nem ismerik más szektorok ugyanezen információit, valamint a b_0^j-t. A központ viszont a b_0^j-t ismeri, de nem ismeri a szektorok technológiáit reprezentáló g_i függvényeket. (Egyelőre ne tegyünk semmi kikötést a g_i és c_i tulajdonságairól.)

A tervezési folyamatban a központ feladata a b_0^j erőforrások olyan felosztása amely felosztás a legnagyobb össznyereséget hozza. (Ha a célfüggvény nyereségmaximálás.) A központ ezt a felosztást kétféleképpen szervezheti:

1. közvetlen korlátfelosztással,
2. közvetett korlátfelosztással.

Tekintsük a két esetet.

1. A közvetlen korlátfelosztás esetén a központ a b_0^j egy lehetséges allokációját készíti el:

$$\sum_{i=1}^{K} b_i^j = b_0^j \qquad (j = 1, \ldots, m) \tag{3.2}$$

A b_i^j értékek ismeretében a következő feladat fogalmazható meg szektorszinten:

$$g_i^j(x_i) \leq b_i^j \qquad (j = 1, \ldots m) \tag{3.3}$$

$$c_i(x_i) \to \max$$

A szektor tehát csupán saját nyereségének maximálására törekszik az adott lehetőségek mellett. A (3.3) feladatok megoldásával megkapjuk az $\hat{x}_i(b_i)$ primál és $\hat{u}_i(b_i)$ duál megoldásokat. A (3.1) globális optimuma akkor érhető el, ha b_0^j felosztása optimális. Ennek az a feltétele, hogy az erőforrások árnyékárait reprezentáló $\hat{u}_i(b_i)$ értékek egyenlők legyenek minden szektorban, azaz

$$\hat{u}_1(b_i) = \hat{u}_2(b_i) = \ldots = \hat{u}_K(b_i). \tag{3.4}$$

Ellenekző esetben a központ új felosztást csinál. A fenti egyszerűsített iterációs folyamat tehát olyan tervezési rendszert képvisel, ahol a központ utasítások formájában közli a szektorokkal, hogy milyen keretek mellett maximálják nyereségüket, a szektorok pedig hatékonysági mutatókat jelentenek vissza a központnak. (Feltételezzük torzításmentesen.)

A kérdés most már az, hogyan algoritmizálható egy ilyen iteratív folyamat. Itt kapcsolódunk a dekompozíciós eljárások irodalmához. Az információáramlás szempontjából ilyen folyamat megvalósításához szolgáló algoritmusok a ten Kate [13], a Kornai—Lipták [16] és a Weitzman [24] által kidolgozott eljárások.

A másik felvetődő kérdés arra vonatkozik, vajon az árnyékárak egalizálódása esetén tényleg elértük-e a globális optimumot. Sajnos ez csak szigorúbb kikötések mellett igaz (konvexitás), míg egyéb esetben csak a globális optimum szükséges feltételeit teljesítettük (l. Kuhn—Tucker-tétel).

2. A közvetett korlátfelosztás vagy árnyékármegállapító eljárás esetén a központ határozza meg az erőforrásokért fizetendő árat $\hat{u}^j$ és megküldi a szektoroknak. A szektorok most a következő feladatot oldják meg:

$$\max\ v_i = c_i(x_i) - \sum_j \hat{u}^j g_i^j(x_i) \tag{3.5}$$

A $c_i(x_i)$ szektornyereség és az igénybevett erőforrásokért fizetett költség különbsége a maximalizálandó függvény. A feladat megoldásával az $x_i(\hat{u}^j)$ megoldásokat kapjuk. A központi feladat globális optimumát akkor érjük el, ha a központ olyan árakat határoz meg, hogy az alábbi két feltétel teljesül:

$$\text{megengedettség: } \sum_i g_i^j[\hat{x}(u^j)] \leq b_0^j \tag{3.6}$$

$$\text{optimalitás: } u^j\{b_0^j - \sum_i g_i^j[\hat{x}_i(u^j)]\} = 0 \tag{3.7}$$

A közvetlen felosztáshoz hasonlóan itt is rokonságot találunk a dekompozíciós algoritmusokkal. A klasszikus Dantzig—Wolfe-algoritmuson kívül egyéb algoritmusok is ilyen folyamatként interpretálhatók (pl. Rosen [22] és az Abadie—Williams [1]).

Sajnos ezúttal is ugyanazok a kérdések problematikusak, amelyek az előző pontban is problematikusak voltak. Csak lineáris esetben igaz, hogy a (3.6) és (3.7) feltételeket kielégítő szektoroptimumok egyúttal a kiinduló feladat globális optimumát adják.

Bevezetőnkben már említettük, hogy a dekompozíciós módszereket alapvetően a nagyméretű feladatok megoldásának igényéből kiindulva dolgozták ki. Az elmúlt években egyre több cikk foglalkozott a különböző módszerek közgazdasági interpretációjával is, a módszereknek a decentralizált tervezési folyamatok elméletével való összekapcsolásával [19]. E két felhasználási irány sajátságos fejlődési utat tett meg. A harmadik generációs gépek megjelenése

a számítástechnikai nehézségek nagyfokú mérsékléséhez vezetett. Ebből következőleg a számítástechnikai felhasználás igénye is csökkent. Ezen túlmenően az is bebizonyosodott, hogy a nagyméretű feladatok dekompozíciós algoritmussal való megoldás több, előre nem várt nehézséget támaszt. Gyümölcsözőnek látszanak viszont az algoritmusokban felismerhető megfontolások a nem lineáris és egyéb nehezebben kezelhető feladatok megoldási lehetőségeinek bővítésében [25].

Egy másik felhasználási terület az elmúlt egy-két évben egyre jobban kirajzolódik. A további kutatások szempontjából érdekesnek látszik a dekompozíciós eljárások kedvező, előnyös tulajdonságainak a kontrollelméletben való felhasználása. Az elmúlt években megjelent cikkek közül kiemeljük Dantzig [7] és Dantzig—Van Slyke [7a] munkáit. Ugyancsak további kutatásokra serkentő gondolat a dekompozíciós módszereknek a gazdasági rendszerelmélettel való összekapcsolása [15].

IRODALOM

1. Abadie, J.—Williams, A. C.: Dual and parametric methods in decomposition. In Graves, R. L.—Wolfe, P. (szerk.): Recent advances in mathematical programming. New York, 1963. McGraw-Hill.
2. Balas, E.: An infeasibility-pricing method for linear programs. Operations Research Vol. 14. No. 5. pp. 847—873.
3. Balinski, M. L.: On some decomposition approaches in linear programming. In Recent Mathematical Advances in Operations Research. Engineering Summer Conferences. University of Michigan. 1964.
4. Charnes, A.—Cooper, W. W.—Kortanek, K. O.: On the theory of semi-infinite programming and a generalization of the Kuhn-Tucker saddle-point theorem for arbitrary convex function. Naval Research Logistics Quarterly Vol. 16. No. 1. (1969) pp. 41—51.
5. Dantzig, B. G.—Wolfe, P.: Decomposition principle for linear programs. Operations Research. January-February 1960. pp. 101—111; Econometrica Vol. 29. No. 4. pp. 767—778.
6. Dantzig, B. G.: Large-scale linear programming. In Dantzig, G. B.—Veinott, A. F. (eds.): Mathematics of the decision sciences. Part I. Americal Mathematical Society 1968.
7. Dantzig, B. G.: Linear control processes and mathematical programming. SIAM Journal on Control. February, 1966. pp. 56—60.

7a. Dantzig, B. G.—Van Slyke, R. M.: Generalized linear programming. [25]-ben, 75—117. o.

8. Dantzig, B. G.: Linear programming and extensions. Princeton, N. J., 1963. Princeton University Press.
9. Dantzig, B. G.: Upper bounds, secondary constraints and block triangularity in linear programming. Econometrica 23. (1955) No. 2. pp. 174—183.
10. Ford, L. R.—Fulkerson, D. R.: A suggested computation for maximal multicommodity network flows. Management Science. October, 1958. pp. 97—101.
11. Geoffrion, A. M.: Elements of large-scale mathematical programming. Management Science. July, 1970. pp. 651—692.
12. Gomory, R. E.: Large and nonconvex problems in linear programming. Proceedings of symposia in applied mathematics. Vol. XV. (1963) American Mathematical Society. pp. 125—139.
13. A. ten Kate: Decomposition of linear programs by direct distribution. Netherlands School of Economics. Rotterdam, 1970. (kézirat, megjelenik az Econometrica-ban)
14. Kelley, J. E.: The cutting-plane method for solving convex programs. Journal of the Society for Industrial and Applied Mathematics. Vol. 8. No. 4. (1960) pp. 703—712.
15. Kornai J.: Gondolatok a többszintű tervezési rendszerekről. Közgazdasági Szemle. 1971/9. sz.
16. Kornai J.—Lipták T.: Two-level planning. Econometrica. Vol. 33. No. 1. (1965) pp. 141—169.

17. KREKÓ B.: Lineáris programozás. Budapest, 1965. Közgazdasági és Jogi Könyvkiadó.
18. KÜNZI, H. P.—TAN, S. T.: Lineare Optimierung grosser Systeme. Lecture Notes in Mathematics. Berlin—Heidelberg—New York, 1966. Springer-Verlag.
19. MALINVAUD, E.: Decentralized procedures for planning. In MALINVAUD, E.—BACHARACH, M. O. L. (eds.): Activity analysis in the theory of economic growth. New York, 1967. Macmillan Company.
20. ORCHARD-HAYS, W.: Advanced linear programming computing techniques. New York, 1968. McGraw-Hill.
21. PREKOPA, A.—MAJTHAY, A.: Nagyméretű lineáris programozási feladatok megoldási módszerei. Kézirat, 1970.
22. ROSEN, J. B.: Primal partition programming for block diagonal matrices. Numerische Mathematik Vol. 6. (1964) S. 250—260.
23. SIVÁK J.—LIGETI, I.: Nagyméretű lineáris programozási feladatok megoldásában felhasználható néhány dekompozíciós eljárás. OT Tervgazdasági Intézet Tájékoztatója. 1969.
24. WEITZMAN, M.: Iterative multi-level planning with production targets. Econometrica. 1970. 1. sz.
25. WISMER, D. A.: Optimization methods for large-scale systems. McGraw-Hill Book Company. New York, 1971.
26. BENDERS, J. F.: Partitioning procedures for solving mixed-variabres programming problems. Numerische Mathematik. Vol. 4. 1962. pp. 239—252.