Stahl János

Egy nagyméretü LP-feladat megoldásáról (I.)

(Mit és hogyan szeretnénk megoldani?)

Bevezetés

Gyakorlati feladatok megoldását ismertető publikációk, röviden esettanulmányok, rendszerint terjedelmesek, ezért sokszor jelennek meg több részben. Maga a gyakorlati probléma, a probléma matematikai modellje, a modell megoldása, annak géprevitele olyan egymástól elválaszthatatlan területeket jelentenek, melyek mindegyikét érinteni kell, és ez már önmagában jelentékeny alsó határt ad egy ilyen cikk terjedelmére. Ezt az említett területek valamelyikéhez kapcsolódó, szokásos értelemben vett eredmények közlése — eléggé nehezen kézbentartható módon — növeli tovább.

Nem minden esetben lehet vagy legalább is érdemes ugyanis ezeket külön publikációba foglalni. (Tisztán matematikai eredmények esetén pl. sokkal egyszerűbb a helyzet: a folyóirat, a szerző, valamint a téma elég jól behatárolja egy-egy publikáció terjedelmét.) Éppen ezért sem szokatlan, hogy ilyen publikációk több részletben jelennek meg és most is mindaz, ami következik, csak az első része valaminek.

A címben szereplő I.-nek azonban most más az igazi oka és erre utal az alcím. Úgy érezzük, hogy a munka jelenlegi fázisában már lehetséges és érdekes megírnunk, hogy mit akarunk és hogyan szeretnénk ezt megcsinálni. Ezt majd szembe lehet állítani a II. résszel, azaz azzal, hogy mi is történt. Minthogy jelen esetben az I. és II. rész megírása között mindenképpen történik még egy és más, ez az említett szembesítést bizonyára érdekesebbé teszi. Mindenesetre, az is egy lehetséges és értékelhető alternatíva, ha a II. rész sohasem születik meg.

Jelen cikkünk 1. fejezetében a megoldandó problémát foglaljuk össze. A 2. fejezetben megfogalmazzuk ennek matematikai modelljét, és röviden ismertetünk egy, a megoldásra kínálkozó dekompozíciós eljárást. A 3. fejezet ennek egy változatával foglalkozik. E változatnak megfelelő számítógépi programok készültek végül is el és ezzel kapcsolatos további megjegyzéseket tartalmaz a 4. fejezet. Az eljárás várhatóan csak közelítő megoldást ad. A közelítő megoldás és az eredeti feladat kapcsolatával foglalkozik a cikk 5. fejezete.

1. A megoldandó probléma

Feladatunk egy kötőelemeket gyártó nagyvállalat adott időszakra (pl. egy negyedévre) vonatkozó maximális fedezettömeget nyújtó termékösszetételének meghatározása.

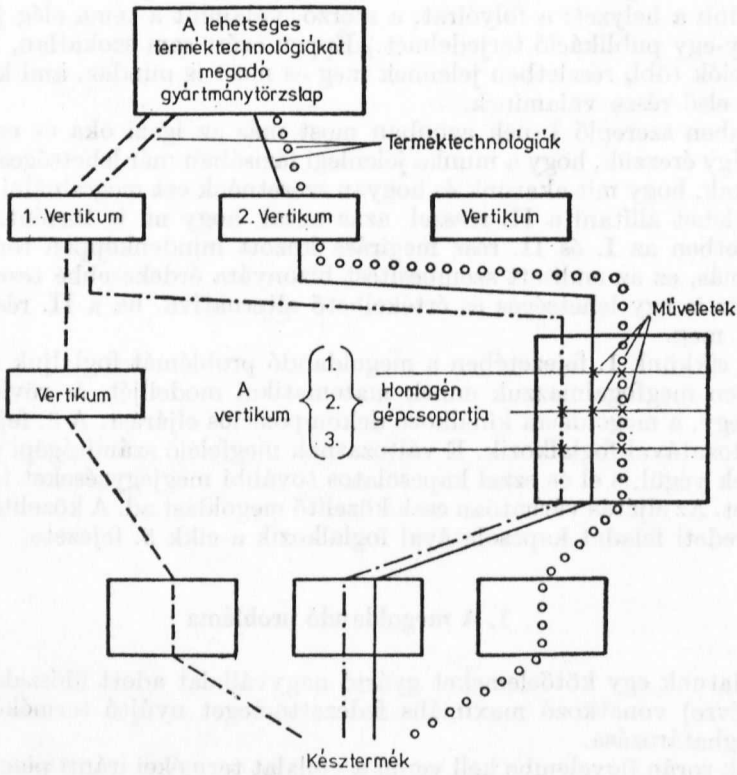
Ennek során figyelembe kell venni a vállalat termékei iránti piaci igényeket (prognózisok, negyedéves tényleges rendelésállomány alapján), valamint a

termékek gyártására vonatkozó kötelező előírásokat (célprogramok, KDD határozatok). Figyelembe kell venni továbbá a homogén gépcsoportok kapacitásadatait, illetve a termékekre vonatkozó alternatív technológiai változatokat.

A vállalat több gyárból áll. Egy-egy gyár homogén gépcsoportjai csoportokba oszthatók és egy ilyen csoportot a továbbiakban vertikumnak nevezünk. Egy-egy vertikum gépcsoportjai azonos jellegű műveleteket végeznek. (A homogén gépcsoport képzés a szokásos módon történik: ha egy művelet elvégezhető a gépcsoport egy tagján, elvégezhető bármely másik tagján is és a műveleti idők aránya minden elvégezhető műveletre ugyanaz.)

A gyártás technológiáját szemlélteti az 1. ábra. Egy termék több technológiai változat szerint gyártható. Egy-egy technológiai változat azt írja elő, hogy egyes műveleteit mely vertikumokban kell végrehajtani. Az már a programozás eredménye, hogy a technológiai változat egy művelete a vertikumot alkotó több, általában termékenként különböző hatékonyságú homogén gépcsoport melyikére kerül. Ugyanis egy terméktechnológia egy vertikumbeli műveletének végrehajtására több homogén gépcsoport is alkalmas és más hatékonysággal. Egy termék egy technológiai változatának műveletei viszonylag kevés vertikumra oszlanak meg.

A szóbajövő termékek száma 10 000 körül van, ezekhez körülbelül 15 000 terméktechnológia tartozik.



1. ábra

Egy terméktechnológia átlagban 4—5 vertikumot érint. A vertikumok száma kb. 50, az összes homogén gépcsoport száma 700. A legnagyobb vertikum 50 homogén gépcsoportot tartalmaz, egy vertikumot maximálisan kb. 500 terméktechnológia érint. Egy-egy negyedéves program várhatóan 5000 terméket és ennek megfelelően kb. 8000 terméktechnológiát vesz figyelembe.

Úgy tervezzük, hogy egy-egy negyedéves optimalizálás négyhónapos időszakot fog át és a tárgynegyedév előtti hónap eleje termelési és anyagleltár adatai, valamint a tárgynegyedévre már beérkezett rendelésállomány alapján történne. A program a tárgynegyedévet megelőző hónap közepére, azaz mintegy két hét alatt készülne el. Ily módon két egymást követő negyedéves optimalizálás programja egy negyedév utolsó hónapjának második felében átfedi egymást. Egy negyedév utolsó hónapjának első felében a termelés a korábbi optimalizálás eredményeinek megfelelően folyna és ennek következményeit a következő negyedévre vonatkozó optimalizálás figyelembe venné. Ugyanígy figyelembe kell venni a korábbi optimalizálás eredményeinek egyéb hátralékos (visszaigazolt, de valamilyen okból elmaradt) részeit. Általában az átfedés folytán korrigálni lehet (és korrigálni kell) a bármilyen operatív okból bekövetkezett eltéréseket.

Az első pillanatban hosszúnak tetsző átfutási idő sok oknak együttes következménye. Remélhetőleg a cikkben sikerül mindezeket világossá tenni. Mindenesetre a cikk 4. fejezetében még visszatérünk erre a kérdésre.

A modell alapján tervezünk éves termékösszetétel optimalizálásokat is. Ezeknél előrebecsült éves rendelés-állomány és éves kapacitásadatok alapján dolgoznánk.

A modellhez elég nagyméretű alapadat-előkészítő és alapadat-karbantartó programrendszer tartozik, amely már elkészült, de ezzel cikkünkben nem foglalkozunk.

2. A probléma matematikai modellje

Vezessük be a következő jelöléseket. Legyen egy adott időszakban

x_{ij} — az i-edik termékből (a termék) j-edik technológiájával előállított mennyiség;

x_{ijkl} — a fenti mennyiségnek az a része, amelynek a k-adik vertikum l-edik homogén gépcsoportján kerül megmunkálásra (ha a j-edik technológia előírja a k-adik vertikum érintését);

t_{ijkl} — az előbbi művelet adott normaideje (azaz az egységnyi termékmennyiség megmunkálásához szükséges idő);

T_{kl} — a k-adik vertikum l-edik homogén gépcsoportján rendelkezésre álló idő;

A_i és F_i — az i-edik termékből előállítandó mennyiségre vonatkozó alsó és felső korlátok;

C_{ij} — az i-edik termékből a j-edik technológiával előállított mennyiségre jutó egységnyi fedezeti tényező.

Az elmondottak alapján a termelési program meghatározását az alábbi lineáris programozási feladat megoldásán keresztül képzeljük

$$A_i \leqq \sum_j x_{ij} \leqq F_i \qquad (i = 1, 2, \ldots)$$

$$\sum'_{(i,j)} t_{ijkl}\, x_{ijkl} \leqq T_{kl} \qquad \text{(a } k\text{-adik vertikum homogén gépcsoportjaira)}$$

$$x_{ij} - \sum'_l x_{ijkl} = 0 \qquad \text{(a } k\text{-adik vertikumot érintő technológiákra)} \tag{2.1}$$

$$(k = 1, 2, \ldots)$$

$$x_{ij},\, x_{ijkl} \leqq 0$$

$$\max \left(\sum_{ij} c_{ij}\, x_{ij} \right)$$

ahol a Σ' jellel arra figyelmeztetünk, hogy az összegzés azon indexekre terjed ki, melyeknek megfelelő változók a megfelelő vertikummal kapcsolatban vannak. (Tulajdonképpen eleve csak létező (ij), (kl) és $(ijkl)$ kombinációknak megfelelő változókat definiáltunk és a továbbiakban nem is fogjuk a Σ' jelet használni.)

Az említett méretek alapján lehetetlennek látszik a (2.1) feladat közvetlen megoldása. Egyrészt ezért, másrészt pedig azért, mert rögzített x_{ij}-k mellett a második feltételcsoport egymástól független részfeladatokra esik szét — egy-egy vertikum egy-egy egyszerűnek tekinthető feladatot határoz meg — kézenfekvőnek látszik egy [2]-nek megfelelő dekompozíció használata.

Ezen eljárás szerint kiindulunk az

$$A_i \leqq \sum_j x_{ij} \leqq F_i \qquad (i = 1, 2, \ldots)$$

$$x_{ij} \geqq 0$$

$$\max \left(\sum_{ij} c_{ij}\, x_{ij} \right)$$

feladat megoldásából.

Általában, az eljárás egy lépésében először rögzített $\tilde{x}_{ij}$-k mellett minden k-ra megoldjuk a

$$\sum_{ij} t_{ijkl}\, x_{ijkl} \leqq T_{kl} \qquad (l = 1, 2, \ldots)$$

$$\sum_l x_{ijkl} = \tilde{x}_{ij} \qquad (i, j = 1, 2, \ldots) \tag{2.2}$$

$$x_{ijkl} \geqq 0$$

vertikumfeladatokat.

Ha ezen feladatok mindegyikének van megoldása, készen vagyunk. Az $\tilde{x}_{ij}$-k a vertikumfeladatokból most meghatározott x_{ijkl}-lekkel együtt a (2.1) feladat optimális megoldását szolgáltatják.

Ha van olyan vertikumfeladat, melynek nincs lehetséges megoldása, akkor minden ilyen k-ra a Farkas lemma folytán van [és a (2.2) feladatok megoldására választott módszer segítségével többnyire adódik is] olyan (p_{kl}, q_{ijk})-rendszer, hogy minden $(ijkl)$-re fennáll

$$p_{kl}\, t_{ijkl} - q_{ijk} \geqq 0$$

és teljesül a

$$\sum_l p_{kl}\, T_{kl} - \sum_{(ij)} q_{ijk}\, \tilde{x}_{ij} < 0$$

egyenlőtlenség.

A lépés második felében új $\tilde{x}_{ij}$-ket határozunk meg. A (2.2) feladatok megoldhatóságát elérendő a már rendelkezésünkre álló feltételeket tartalmazó ún. központi feladatot bővítjük a

$$\sum_{(ij)} q_{ijk}\, x_{ij} \leqq \sum_{l} p_{kl}\, T_{kl}$$

feltételekkel, ahol az együtthatók az előbb meg nem oldhatónak bizonyult vertikumfeladatokból adódtak. Új $\tilde{x}_{ij}$-ket olymódon kapunk, hogy a korábbi és most meghatározott feltételek mellett maximalizáljuk $\sum_{ij} c_{ij}\, x_{ij}$-t.

Az új $\tilde{x}_{ij}$-k birtokában a vertikumfeladatok megoldásával folytatjuk az eljárást, mely a most elmondott lépések végesszámú végrehajtása után véget ér. [Ha valamikor olyan eset lépne fel, hogy az $\tilde{x}_{ij}$-kre vonatkozó feladatnak nincs lehetséges megoldása, nincs lehetséges megoldása a (2.1) feladatnak sem.]

Az eljárás így elvben jó, a gyakorlatban azonban minden bizonnyal nem, valószínűleg túl időigényes. Semmilyen alapunk sincs ennél erősebben fogalmazni.

3. A feladat megoldására tervezett eljárás

Ebben a fejezetben az előző fejezetbeli dekompozíciós eljáráson végrehajtott változtatásokkal és a változtatások okaival foglalkozunk. Mindez már a számítógépes realizációval is összefonódik.

Először az $\tilde{x}_{ij}$-k meghatározására szolgáló központi feladatot vizsgáljuk. Ilyen eljárások során, mondhatni, szokásos, hogy az $\tilde{x}_{ij}$-knak az eljárás során bekövetkező változását minél jobban korlátozzák, de legalább is nem engedik meg, hogy a központi feladat nyilvánvalóan rossz $\tilde{x}_{ij}$ értékeket adjon. Ezért a kiinduláskor az $\tilde{x}_{ij}$-kre vonatkozó

$$A_i \leqq \sum_{j} x_{ij} \leqq F_i$$

feltételek mellé további feltételeket vezetünk be. Ezek a pótlólagos kezdeti feltételek két feltételcsoportból tevődnek össze.

Az első feltételcsoport feltételeit azon vertikumok adják, melyek egyetlen homogén gépcsoportot tartalmaznak. Ilyen k-ra $x_{ijkl} = x_{ij}$ és a megfelelő

$$\sum_{(ij)} t_{ij}\, x_{ij} \leqq T_k$$

feltételt a központi feladatban szerepeltetjük.

A másik feltételcsoport elemei a további vertikumok alapján adódnak. Egy-egy (2.2) vertikumfeladat megoldhatóságához szükséges, hogy az x_{ij}-k meghatározta termelési program beleférjen a vertikumbeli gépek kapacitásába akkor, ha minden gépre a számára — a t_{ijkl}-ek szempontjából — legkedvezőbb termék kerül. Ugyancsak szükséges, hogy a szóbanforgó termelési program úgy is beleférjen a gépkapacitásba, ha az x_{ij} mennyiség a terméktechnológia szempontjából legkedvezőbb gépre kerül. Tehát minden több homogén gépcsoportot tartalmazó vertikum esetén induláskor bevezetünk a központi feladatba egyrészt egy

$$\sum_{(i,j)} x_{ij} \leqq \sum T_{kl} / \min_{(i,j)} t_{ijkl}$$

alakú, másrészt egy

$$\sum_{(i,j)} (\min_{l} t_{ijkl})\, x_{ij} \leq \sum_{l} T_{kl}$$

alakú feltételt. [Általában nem lehet megmondani, hogy a két feltétel közül melyik az erősebb. Vannak olyan vertikumok, ahol a vertikum minden l homogén gépcsoportjára $t_{ijkl} = t_{kl}$, azaz az egységnyi műveletidő független a terméktől, illetve a terméktechnológiától. Ilyen esetben a második feltétel nyilván felesleges, illetve a (2.2) vertikumfeladat nem ún. általánosított hanem közönséges szállítási feladat.]

Az $\tilde{x}_{ij}$-k változásának korlátozását az előbb szokásosnak neveztük. Nyilvánvalónak tekintik ugyanis, hogy egy optimálishoz közeli $\tilde{x}_{ij}$-rendszerből kiindulva kevesebb lépésben — a központi feladat és a vertikumfeladatok megoldásának kevesebb számú megismétlésével — lehet az optimális $\tilde{x}_{ij}$-khoz eljutni.

Ezt alá is támasztja több-kevesebb számítógépes tapasztalat. Határesetben nyilvánvaló, hogy optimális $\tilde{x}_{ij}$-kből indulva egyetlen lépésben véget ér az eljárás. Gondoljunk továbbá a Dantzig—Wolfe eljárásra, amelynek az előbbi fejezetben ismertetett eljárás lényegében a duálisa. Ha rendelkezésünkre állnának a közös feltételekhez tartozó duálváltozók optimális értékei (a feladat duálisa optimális megoldásának megfelelő része), a Dantzig—Wolfe eljárás egyetlen lépésének végrehajtásával meg tudnánk oldani a feladatot. A Dantzig—Wolfe eljárással kapcsolatban is van olyan tapasztalat, hogy a közös feltételek multiplikátorait egy „reális" (nyilván az optimálishoz közeli) érték közelében tartva, az eljárás lépéseinek száma lényegesen csökkenthető. [1] (Mindez egyébként némi magyarázatot is ad arra, hogy általában miért működik egy [2]-típusú dekompozíció többnyire sokkal jobban, mint a Dantzig—Wolfe eljárás, holott a két eljárás között — legalább is a lineáris esetben — matematikai szempontból nincs különbség. Mindegyik természetesen a maga esetében működik jól vagy kevésbé jól. Egy nagyméretű feladat alkotóinak általában magáról a feladatról van elképzelésük és nem annak duálisáról. Következésképpen a közös változókról sokkal inkább elképzelhető az, hogy már induláskor optimális értékük köré tudják korlátozni, mint a duális feladat változóiról. Ugyanakkor a Dantzig—Wolfe eljárást éppen a feladat duálisára vonatkozó ismeretek, elképzelések alapján lehetne hatékonyabbá tenni.)

Van azonban az $\tilde{x}_{ij}$-k változása korlátozásának egy később részletezendő további szempontja is. Úgy képzeljük, hogy az eljárást nem érdemes az optimális megoldásig folytatni. Tehát a közbülső $\tilde{x}_{ij}$-rendszereknek is lehetőleg elfogadhatóknak kell lenniök.

Az eddigieket összefoglalva, a dekompozíciós eljárás során adódó központi feladatok tehát

$$A_i \leq \sum_{j} x_{ij} \leq F_i \qquad (i = 1, 2, \ldots)$$

$$\sum_{(i,j)} t_{ij}\, x_{ij} \leq T_k \qquad \text{(ha a } k\text{-adik vertikum egyetlen homogén gépcsoportot tartalmaz)}$$

(3.1)

$$\left.\begin{aligned} &\sum_{(i,j)} x_{ij} \leq \sum_{l} T_{kl} / \min_{(i,j)} t_{ijkl} \\ &\sum_{(i,j)} (\min_{l} t_{ijkl}) x_{ij} \leq \sum_{l} T_{kl} \end{aligned}\right\} \quad \text{(ha a } k\text{-adik vertikum több homogén gépcsoportot tartalmaz)}$$

$$\sum_{(i,j)} q_{ijk}^{(n)} x_{ij} \leq \sum_{l} p_{kl}^{(n)} T_{kl} \qquad (n = 1, 2, \ldots;\ k = \ldots)$$

$$\max \left(\sum_{(i,j)} c_{ij} x_{ij} \right)$$

alakúak, ahol az n felső index azt fejezi ki, hogy az utolsó feltételcsoport elemei esetleg különböző lépésekben, különbözőképpen rögzített $\tilde{x}_{ij}$ értékekkel megoldott (2.2) vertikumfeladatok alapján adódtak.

Hogyan oldjuk meg ezeket a feladatokat?

Minthogy a feltételek nagy többségét, kb. 5000-t az egyszerű szerkezetű első feltételcsoport teszi ki, a központi feladat megoldása kézenfekvő eszközének az ún. általánosított felsőkorlátok módszere [3] látszik.

A központi feladat további feltételeinek száma az eljárás kezdetén mintegy 70, ami az eljárás során várhatóan néhány százra emelkedik.

A problémát IBM gépen kellett megoldanunk. Az IBM gépen rendelkezésünkre álló Mathematical Programming System (MPS) nem tartalmazza az általánosított felsőkorlátok módszerét. Egy komplett, általánosított felsőkorlátos módszert is tartalmazó lineáris programozási rendszer elkészítése olyan volumenű munka, hogy ezt a lehetőséget az adott esetben kizárhattuk. Egy további lehetőség lett volna, hogy csináljunk az MPS-re építve egy általánosított felsőkorlátos módszert alkalmazó programot. [Szükségszerű, hogy az MPS-re építsünk. 4—500 feltételes lineáris programozási feladatok megoldásánál már szükség van mindazokra az eszközökre (pl. a numerikus stabilitást biztosító eszközök), melyekkel az MPS rendelkezik. Azt, hogy mindezeket magunk produkáljuk — a nagy programozási ráfordítás miatt — ismét csak nem vállalhattuk.]

Az MPS-hez így hozzányúlni egyrészt elég nehézkes és nem is olcsó, másrészt ezt a lehetőséget túl sok programozási munkát igénylőnek ítéltük, minden bizonnyal jogosan. Végül úgy döntöttünk, hogy a központi feladat megoldását a Dantzig—Wolfe eljárás alapján végezzük olymódon, hogy a (3.1)-beli első feltételcsoportot tekintjük részfeladatnak. Ennek a programját is a MPS-re építettük. Az MPS használata most is nehézkes és drága, de ennek a realizálása legalább viszonylag egyszerű volt.

Az MPS ilyen használata el is tünteti azt a különbséget, mely csoportkorlátos feladatoknál az általános felsőkorlátos technika alkalmazásának javára mutatkozik a Dantzig—Wolfe eljárás alkalmazásával szemben.

Az értelmes megoldás nyilván az MPS Extended valamelyik változatának megszerzése és alkalmazása lett volna, mivel az MPS Extended tartalmazza az általánosított felsőkorlátos módszert. Azonban ez az út sem bizonyult járhatónak. Bár, ha őszinték akarunk lenni, mi sem gondoltuk volna, hogy az MPS ilyen használata ennyire körülményes és drága, vagy hogy a hazánkban üzemelő IBM rendszereken nem áll rendelkezésre egyetlen hatékony és jól kezelhető lineáris programozási szubrutin sem. Mindez — természetesen?! — a dokumentációkból nem derül ki. Minthogy az MPS Extended-hez is csak (IBM) dokumentációk alapján fűzhetünk reményeket, az ember némileg bizonytalan, hogy annak használata lenne-e az igazi (vagy legalább is jobb) megoldás?

Ezen a ponton lehet bevezetni a központi feladat egy további módosítását, mely a központi feladat megoldására választott módszerrel is összefügg. Az eljárás során nem (2.3)-t, hanem az annál gyengébb

$$A_i \leq \sum_j x_{ij} \leq F_i \qquad (i = 1, 2, \ldots)$$

PÓTLÓLAGOS KEZDETI FELTÉTELEK

$$\sum_{(i,j)} q_{ijk}^{(n)} x_{ij} - z_k \leq \sum_l p_{hl}^{(n)} T_{kl} \qquad (n = 1, 2, \ldots;\ k = \ldots) \tag{3.2}$$

$$z_k \leq D_k$$

$$\max \left(\sum_{ij} c_{ij} x_{ij}\right)$$

alakú feladatokat oldjuk meg, ahol a D_k-k rögzített számok.

Ugyanis a Dantzig—Wolfe eljárás alkalmazása során el akarunk kerülni, egy formális indulóprogram kereső első fázist. Itt egyrészt arra gondoltunk, hogy a Dantzig—Wolfe eljárástól azt várjuk, hogy indulóprogram keresésnél az optimum közelébe elég hamar eljusson, másrészt pedig arra, hogy a D_k értékek alkalmas megválasztásával elérhető, hogy egy központi feladatot attól a megoldástól tudjunk folytatni, amelynél a korábbi központi feladat megoldását abbahagytuk. Továbbá, nem is szükséges a (3.1) feladat feltételeit minden esetben pontosan kielégíteni, amivel részletesebben a vertikumfeladatok tárgyalása során foglalkozunk majd.

Ezért egy központi feladat megoldásánál úgy járunk el, hogy először a korábbi központi feladat megoldásából és ennek megfelelő D_k-értékekből kiindulva minimalizálunk egy $\Sigma d_k z_k$ összeget, ahol a d_k-kat a központi feladat megoldása előtt (lényegében a korábbi vertikumfeladatok megoldása alapján) határozzuk meg (és esetleg a feladat megoldása közben annak állapotától függően változtatjuk is). Majd, ha $\Sigma d_k z_k$ érték elég kicsi, a D_k-knak a megfelelő pillanatnyi értékét rögzítve elkezdjük (3.2) megoldását.

(3.2) egy $\tilde{x}_{ij}$-megoldásának (vagy közelítő megoldásának) birtokában kezdjük meg a (2.2) vertikumfeladatok megoldását. Pontosabban, azt kell eldöntenünk, hogy a (2.2) feltételrendszereknek van-e lehetséges megoldásuk vagy sem. Ez a

$$\sum_{(i,j)} t_{ijkl} x_{ijkl} - y_{kl} \leq T_{kl} \qquad (l = 1, 2, \ldots)$$

$$\sum_l x_{ijkl} = \tilde{x}_{ij} \qquad (i, j = \ldots) \tag{3.3}$$

$$y_{kl},\ x_{ijkl} \geq 0$$

$$\max \left(-\sum_l h_{kl} y_{kl}\right)$$

lineáris programozási feladatok megoldásával végezhető el, ahol a h_{kl}-ek rögzített számok (melyek nagysága az iteráció sorszámától is függhet). Az

$$\bar{y}_{kl} = T - y_{kl}$$

változókat bevezetve, ahol T alkalmasan nagy szám, az adódó

$$\sum_{(i,j)} t_{ijkl} x_{ijkl} + \bar{y}_{kl} \leq T_{kl} + T \qquad (l = 1, 2, \ldots)$$

$$\sum_l x_{ijkl} = \tilde{x}_{ij} \qquad ((i, j) = \ldots) \tag{3.4}$$

$$x_{ijkl} \geq 0$$

$$\max \left(\sum_l h_{zl} \bar{y}_{kl}\right)$$

feladat is egy általánosított szállítási feladat. Ennek megoldására az [5]-beli általánosított felsőkorlátos módszert programoztuk be. Ez egy duál algoritmus, amit azért részesítettünk előnyben az eredeti primál változattal [3] szemben, mert úgy képzeltük, hogy ily módon hatékonyan fel tudjuk használni egy vertikumfeladat megoldásánál a vertikumfeladat korábbi vizsgálatánál kapott megoldást.

A vertikumok méretei lehetővé teszik, hogy a közbülső megoldások, illetve optimális megoldás megőrzésétől eltekintve, mindent az operatív memóriában oldjunk meg. Ugyanakkor az eljárásba szinte pontosan annyi eszközt (pivotkiválasztási, újrainvertálási kritérium, újrainvertáló szubrutin) kellett végül beleépíteni, mint egy általános lineáris programozási program (rendszer) optimalizáló részébe. Ez tulajdonképpen nem is meglepő, mert egy általánosított szállítási feladat ilyen szempontból semmivel sem speciálisabb, mint egy tetszőleges lineáris programozási feladat.

A próbaszámítások eredményei elég meglepőek voltak. Már említettük, hogy egyes vertikumok esetén a (2.2) feltételrendszer szállítási feladatra redukálódik. Szállítási feladat megoldására már korábban kidolgoztunk egy több esetben is hatékonynak bizonyult programot. (Ez kb. azt jelenti, hogy nagyméretű szállítási feladatokat is gyorsan megoldott.) A szóbanforgó néhány vertikumfeladat kezelésére ennek IBM változatát készítettük el. A próbaszámítások kb. azt adták, hogy az előbbi program nagyjából ugyanannyi idő alatt old meg egy általánosított szállítási feladatot, mint utóbbi egy hasonló méretű közönséges szállítási feladatot. Nyilván nem a méret az egyetlen tényező, mely egy feladat komplikált voltát meghatározza (nagyon lényeges pl. a megengedett gépcsoporttechnológia párok száma és eloszlása), de az eredményeket mégis úgy tekintjük, hogy a szállítási feladat programjának finomítása is jelent egy kevés tartalékot.

Valójában nem (3.3)-mal, illetve (3.4)-gyel, hanem ezek egy tovább módosított változatával foglalkozunk. Elvileg akkor fejezhetjük be az eljárást, ha a központi feladat megoldása alapján képzett valamennyi vertikumfeladatnak van lehetséges megoldása. Gyakorlati szempontból nyilván az is kielégítő, ha valamennyi vertikumfeladat megoldható, vagy majdnem megoldható. Ugyanis egy homogén gépcsoport kapacitásának bizonyos mértékű túllépése elfogadható, mivel egyszerűen értelmezhető az egymást követő negyedéves programok átfedése folytán.

Ezért a vizsgált vertikumfeladatok

$$\sum_{(i,j)} t_{ijkl}\, x_{ijkl} - \bar{y}_{kl} - \bar{\bar{y}}_{kl} \leq T_{kl}$$

$$\sum_{l} x_{ijkl} = x_{ij}^{(n)} \tag{3.5}$$

$$\bar{y}_{kl} \leq 0{,}05\, T_{kl}$$

$$\bar{y}_{kl},\ \bar{\bar{y}}_{kl},\ x_{ijkl} \geq 0$$

$$\max \left(-\sum_{l} \bar{y}_{kl}\, \bar{h}_{kl} - \sum_{l} \bar{\bar{h}}_{kl}\, \bar{\bar{y}}_{kl}\right)$$

alakúak. Azaz, a vertikum minden homogén gépcsoportján az esetleges T_{kl}-en felüli gépidőfelhasználást két részre osztottuk. Az elsőt $\bar{y}_{kl}$ méri és ez nem lehet több, mint T_{kl} 5%-a, a második részt adja meg $\bar{\bar{y}}_{kl}$. Ezen utóbbiak $\bar{\bar{h}}_{kl}$ célfügg-

vény-együtthatóit úgy választottuk, hogy bármelyik $\bar{y}_{kl}$ változó preferált legyen bármelyik $\bar{\bar{y}}_{kl}$ változóhoz képest. A (3.5) feladatok is nyilván átalakíthatók általánosított vagy közönséges szállítási feladattá.

A 0,05% kb. egy hetet jelent és ilyen túllépések az egymást követő negyedéves programok átfedése folytán elfogadhatók. [Formálisan is felírható az y-változók és a (3.2)-beli z-k kapcsolata. Mindenesetre, ezért nem feltétlenül „kell" a (3.1)-beli utolsó feltételeket pontosan kielégíteni.]

A programrendszert úgy készítettük el, hogy egy-egy iterációs lépés végén lehetőség van egyes vertikumfeladatok újravizsgálatára. Erre akkor lehet szükség, mikor egy vertikumfeladat eredményei alapján úgy látszik, hogy más h-értékek esetén kedvezőbb, vagy elfogadhatóbb gépidőtúllépések adódnak. Ez az előbbiek szerint nem gépidőigényes, ugyanakkor az újrafuttatásból a központi feladat számára adódó feltétel pontosan úgy használható, mint a vertikumfeladat előző megoldásából kapott.

4. A programrendszerre vonatkozó néhány további megjegyzés

Az előző fejezetben már foglalkoztunk a dekompozíciós eljárás egyes részeinek géprevitelével. Már ezeknél is figyelembevettük az egész rendszer működésére vonatkozó elképzeléseinket, illetve helyenként utaltunk is erre. Mindenesetre, ebben a fejezetben először is megpróbáljuk az egész rendszer működésére vonatkozó elvárásainkat összefoglalni.

Az eljárást, illetve annak egy lépését a 2. ábra szemlélteti. 50 vertikumot feltételezve, melyekből 10 egy homogén gépcsoportot tartalmaz, 15 pedig szállítási feladatként kezelhető, a pótlólagos kezdeti feltételek száma 70 körül van.

A Dantzig—Wolfe eljárással kezelt központi feladat feltételeinek a száma minden lépésben kb. 40-nel nő. (Kezdetben a vertikumfeladatok között alig van megoldható, későbbi lépésekben feltételezhetően egyre több oldható meg, de ugyanakkor nőhet azon vertikumok száma is, melyeket egy lépésben többször is megvizsgálunk és ezek a központi feladat új feltételeinek a számát növelhetik.) Tehát a Dantzig—Wolfe eljárás extremális feladata feltételeinek a száma várhatóan így alakul 70, 110, 150, . . .

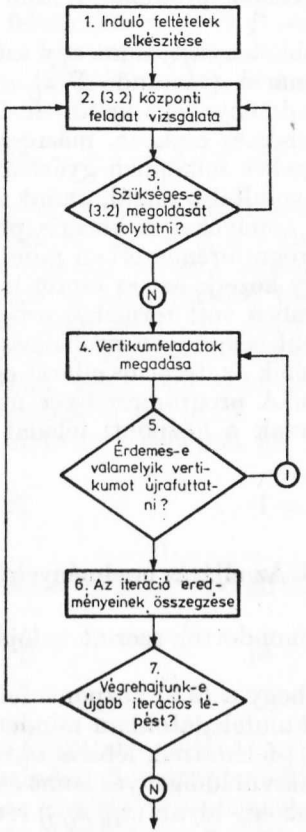
A programot úgy írtuk, hogy a terméktechnológiákra vonatkozó csoportkorlátok meghatározta részfeladatnak az eljárás során adódó valamennyi megoldását megőrizzük és az ezeknek megfelelő változók végig szerepelnek minden központi feladat extremális feladatában. Ez meglehetősen sok közbülső eredmény adminisztrálását kívánja, ami eléggé memória és időigényes. Viszont az a reményünk, hogy így lényegesen redukálódik a központi feladatok megoldásánál szükséges Dantzig—Wolfe lépések száma és így időben végül is nyerünk. Mindenesetre reméljük, hogy a központi feladatok megoldásánál 100, 60, 60 . . .-nál kevesebb számú Dantzig—Wolfe lépéssel célt érünk.

Továbbá 8000 olyan terméktechnológiával számolva, melyeket egy optimalizálás során figyelembe kell vennünk, egy Dantzig—Wolfe iterációs lépés 2,5 perc körül van a SZÜV IBM/370 rendszerén. Így a központi feladatok megoldására fordított idő 10 iterációs lépéssel számolva 22—26 óra körül lehet.

Egy-egy központi feladat megoldását úgy képzeljük, hogy a legrosszabb esetben a 30. lépés után kell az eredeti feladat célfüggvényét maximalizálnia. Ennek ellenőrzésére, illetve biztosítására különféle eszközöket építettünk a programba. Végső eszközt a futás megszakítása, illetve újraindítása jelent,

amit a séma 3. blokkja jelez. [Ebben az esetben mód van a (3.2) feladat d és D értékeinek megváltoztatására is.]

Egy-egy vertikumfeladat megoldására átlagban 4 percre lehet szükség az említett rendszeren. Ez lépésenként 160 perc, összesen kb. 22—26 óra. A séma 5. blokkja a vertikum feladatoknál említett esetleges újrafuttatás(oka)t jelöli.



2. ábra

A séma utolsó blokkja az iterációs lépés eredményét összegzi. Ennek, valamint a korábbi hasonló eredmények alapján dönthetünk az eljárás befejezéséről vagy újabb lépés végrehajtásáról.

Az, hogy kb. 10 iterációs lépés elég lesz, a [2]-dekompozícióra vonatkozó kedvező tapasztalatoknak [4], [6] talán túl merész általánosítása. De elég biztosnak látszik, hogy ennyi iterációs lépés valamelyike a vállalat számára olyan akceptálható hasznos eredményt ad, amely még a ráfordításokkal is arányban áll. Ebben segítenek az említett beavatkozási pontok is.

Elvben a rendszer automatikussá tehető. Ez azonban nem látszik célszerűnek és valójában nem is szükséges. Az első néhány lépés elvégezhető minden beavatkozás nélkül, a továbbiaknál szükséges beavatkozáshoz a rendszer

elegendő információt ad. Úgy látszik, hogy meg lehet majd tanítani a vállalat szakembereit arra, hogy ezeket az információkat miként használják.

Ha mindezen várakozásaink realizálódnak és a rendszer élni fog, a kapott végeredmény nem rossz. (Az más kérdés, hogy ekkor érdemes-e és milyen mértékben érdemes a rendszer egyes részeit, pl. a központi feladat kezelését javítani.) Végül is az említett ráfordítással sikerülne 10—12 nap alatt egy negyedéves termelési programot produkálni, ami a jelenlegi rendszerhez képest nagy lépés lenne előre.

Valójában a rendszer többet nyújt, mint egy szűkebb értelemben vett termelési programot, hiszen már a gyártandó $\Sigma_i x_{ij}$ mennyiségek is a programozási feladat megoldása eredményeként adódnak. Tehát a modell egyrészt a (negyedéves vagy éves) tervezés eszköze, másrészt azt is megoldja, hogy a tervezett termékmennyiségeket miképpen gyártsák le.

Maga a probléma is bonyolultabb annál, amint azt 2.-ben leírtuk. Ismertek továbbá azok a kritikák, amelyeket a lineáris programozás, mint tervezési eszköz kapott. Az egész programrendszerben mindössze néhány utasítást kell megváltoztatni ahhoz, hogy kezelje azt az esetet is, amikor a $\Sigma_j x_{ij}$-k rögzítettek; ekkor szűkebb értelemben vett termelési program meghatározásáról van szó. (Az végképp nem jelent semmilyen problémát, ha adott x_{ij}-khez tartozó x_{ijkl}-eket keresünk: ez történik az iterációs eljárás egyes lépéseiben a vertikumfeladatok megoldásakor is. A programrendszer úgy készült, hogy ez végrehajtható tetszőleges, nemcsak a központi feladat megoldásából adódó x_{ij}-k esetén is.)

5. Az eljárás eredményeiről

Az előző fejezetekben elmondottak szerint valójában nem feltétlenül oldjuk meg a (2.1) feladatot.

Elvileg elérhető ugyan, hogy a (3.2) központi feladatban minden z-változót és a (3.3) vagy (3.5) vertikumfeladatokban minden y-változót zérusra szorítsunk le, amennyiben a (2.1)-feladatnak létezik optimális megoldása. Ez azonban minden bizonnyal rendkívül időigényes lenne és erre valójában szükségünk sincs. Amire szükség van, az egy olyan (x_{ij}, x_{ijkl}) rendszer, mely a (2.1) feladat célfüggvénye és feltételei, valamint a megbízó ráfordításai szempontjából a megbízó számára akceptálható. Ennek eldöntéséhez viszont segítséget nyújt, ha tudunk valamit mondani az eredeti feladat optimális megoldásáról, illetve annak a feladatnak az optimális megoldásáról, melyre vonatkozóan a kapott eredmény lehetséges megoldás. (Ugyanis egyes homogén gépcsoportokon az eredmény gépidőfelhasználása várhatóan meghaladja a megadott T_{kl}-értékeket.)

Ezzel kapcsolatos a következő néhány egyszerűen igazolható állítás.

Legyen a $(v, w_k^{(n)}, w_k)$ rendszer egy (3.2) központi feladatnak Dantzig—Wolfe-féle megoldása során az extremális feladat valamelyik megoldásánál nyert multiplikátor-rendszer. [v a pótlólagos kezdeti feltételekhez tartozó multiplikátorokból áll, $w_k^{(n)}$ egy

$$\sum_{(i,j)} q_{ijk}^{(n)} x_{ij} - z_k \leq \sum_l p_{kl}^{(n)} T_{kl}$$

feltételhez, w_k pedig a

$$z_k \leq D_k$$

feltételhez tartozó multiplikátor.]

Akkor az $(u_i, v, w_k^{(n)}, w_k)$ rendszer, ahol

$$u_i = \max_j \left(c_{ij} - v_{ij} - \sum_{k,n} w_k^{(n)} q_{ijk}^{(u)}\right)$$

és t_{ij} a pótlólagos kezdeti feltételekben az (i, j) terméktechnológiához tartozó együtthatókból álló vektor, lehetséges megoldása (3.2) duálisának.

Az ehhez tartozó (duál) célfüggvényérték és a (3.2) aktuális megoldásához tartozó célfüggvényérték különbsége

$$C_1 = \sum_{i,j} [(c_{ij} - v_{ij}) - \sum_{k,n} w_k^{(u)} q_{ijk}^{(u)}] \tilde{x}_{ij} - u$$

ahol u a Dantzig—Wolfe eljárás extremális feladatában szereplő konvexitási feltételhez tartozó multiplikátor, $\tilde{x}_{ij}$ pedig a Dantzig—Wolfe részfeladatnak a szóbanforgó multiplikátorokhoz tartozó extremális megoldása. Tehát a szóbanforgó duális megoldás, a hozzátartozó (duál) célfüggvényérték, ami (3.2) optimumértékének becslése, és végül ennek a becslésnek pontosságát megadó fenti kifejezés adódik a Dantzig—Wolfe eljárás alkalmazása során, ami útmutatást nyújt abban, hogy egy (3.2) központi feladat megoldását meddig érdemes ezzel az eljárással folytatni.

Az $(u_i, v, w_k^{(n)})$ rendszer nyilván megoldása az

$$A_i \leq \sum_j x_{ij} \leq F_i \qquad (i = 1, 2, \ldots)$$

PÓTLÓLAGOS KEZDETI FELTÉTELEK

$$\sum_{(i,j)} q_{ijk}^{(n)} x_{ij} \leq \sum_l p_{kl}^{(n)} T_{kl} \qquad (n = 1, 2, \ldots;\ k = \ldots)$$

$$\max \left(c_{ij} x_{ij}\right)$$

feladat duálisának és így a hozzátartozó (duál) célfüggvényérték [ami ugyancsak rendelkezésünkre áll minden (3.2) központi feladat megoldásának minden lépésében] felső becslése az eredeti (2.1) feladat optimumértékének. [Ez a felső korlát a megfelelő (3.2) feladatra vonatkozó, ugyanazon (Dantzig—Wolfe) lépésben nyert felső korlátnál $\sum_k w_k D_k$-val kisebb.]

Az eljárás mindegyik lépésének befejezésekor (tehát a vertikumfeladatok megoldása után) rendelkezésünkre áll az

$$A_i \leq \sum_j x_{ij} \leq F_i \qquad (i = 1, 2, \ldots)$$

$$\sum_{(i,j)} t_{ijkl} x_{ijkl} \leq T_{kl} + y_{kl} \qquad \text{(a } k\text{-adik vertikum homogén gé-p csoportjaira)}$$

$$x_{kl} - \sum_l x_{ijkl} = 0 \qquad \text{(a } k\text{-adik vertikumot érintő } (i, j) \text{ technológiákra)} \tag{5.1}$$

$$x_{ij},\ x_{ijkl} \geq 0$$

$$\max \left(\sum_{(i,j)} c_{ij} x_i^f\right)$$

lineáris programozási feladat egy megoldása, ahol az eredeti (2.1)-hez képest eltérést jelentő y_{kl}-ek a vertikumok homogén gépcsoportjain ebben a lépésben adódott kapacitástúllépések. Minthogy az $(u_i, v, \sum_n w_k^{(n)} p_{kl}^{(n)}, \sum_n w_k^{(n)} q_{ijk}^{(n)})$ rendszer megoldása a redundáns pótlólagos kezdeti feltételekkel bővített (5.1) feladat duálisának, a hozzátartozó (duál) célfüggvényérték felső becslése az (5.1) feladat optimumértékének. Ezen becslés és az aktuális, a rendelkezésre álló megoldáshoz tartozó célfüggvényérték különbségére

$$C_2 = C_1 - \sum_k w_k D_k + \sum_{kl} p_{kl} y_{kl}$$

adódik, ahol $p_{kl} = \sum_n w_k^n p_{kl}^{(n)}$.

Az eljárás során minden lépésben csak egyszer, az utolsó Dantzig—Wolfe lépésben számítjuk ki C_2-t, ugyanis p_{kl} számítása túl időigényes. [p_{kl} meghatározása akkor lenne egyszerű, ha a

$$\sum_{(i,j)} q_{ijk} y_{ij} \leq \sum_l p_{kl} T_{kl}$$

alakú feltételeket generáló p_{kl} értékeket a feltételekkel együtt tudtuk volna tárolni, ezt azonban nem tudtuk megoldani.]

Valójában tehát az előző részbeli eljárástól azt várjuk, hogy az eljárás lépéseit mintegy 10-szer végrehajtva ki tudunk választani olyan lépést, mikor az (5.1) feladatban az y_{kl}-lek elég kicsik, a $\sum_{i,j} c_{ij} x_{ij}$ célfüggvényérték elég nagy, illetve a hozzátartozó C_2 ehhez képest elég kicsi. A szóbanforgó lépésben x_{ij}-kel, valamint az y_{kl}-ekkel megnövelt T_{kl}-ekkel megoldanánk mégegyszer a vertikumfeladatokat. Ebben az esetben már a minimális gépidőfelhasználás lenne a vertikum-optimalizálás célfüggvénye. (Olyan vertikumok esetén, melyek valamennyi gépen felhasznált gépidő csak a legyártandó darabszámtól függ, azaz a szállítási feladattal modellezhető vertikumoknál, a vertikum-optimalizálás célfüggvénye a vertikum gépcsoportjai közötti valamilyen további preferenciát fejezne ki.)

(Beérkezett: 1976. október 28.)

IRODALOMJEGYZÉK

1. Beale, E. M. L., Hughes, P. A. B., and Small, R. E.: „Experience in using a decomposition program", The Computer Journal, 8 (1965) 13—18.
2. Benders, J. F.: „Partitioning procedures for solving mixed variable programing problems", Numerische Mathematik, 1 (1972) 238—252.
3. Dantzig, G. B. and van Slyke, R. M.: „Generalized upper bounded techniques for linear programming", Journal of Computer and System Sciences, 1 (1967) 213—226.
4. Geoffrion, A. M. and Graves, G. W.: „Multicommodity distribution system design by Benders decomposition", Management Science, 20 (1974) 822—844.
5. Grigoriadis, M. D.: „Dual generalized upper bounding techniques", Management Science, 17 (1971) 269—285.
6. Kovács Á. és Stahl J.: „Dekompozíciós eljárás a szén termelésének és elosztásának optimalizálására", SZIGMA, III (1970) 97—107.

ON THE SOLUTION OF A LARGE SCALE LP PROBLEM (I)

The paper forms the first part of a case study.

Chapter 1 explains the production planning and programming problem to be solved: the product mix of a large screw mill is to be determined by taking technological constraints and orders into account and maximizing profits. In Chapter 2 we describe the mathematical model in the form of a LP. The problem size being large it cannot be attacked directly, thus we discuss also a decomposition procedure for its solution.

In Chapter 3 we deal with the computerization of this procedure, we discuss the reasons, facts and expectations which have been pondered when selecting and programming the algorithms for each step in the decomposition. All these are connected with the formulation and reformulation of the model itself. The conclusions are summarized in Chapter 5. Here we also set up that problem whose solution is given by this procedure and analyse the relation between this problem and the original problem of Chapter 2.

О РЕШЕНИИ КРУПНОЙ ЗАДАЧИ ПО ЛИНЕЙНОМУ ПРОГРАММИРОВАНИЮ

Настоящая статья является первой частью написанного автором очерка.

В первой главе автор излагает задачу по планированию производства и программированию, которую следует решить в случае выпускающего крепежные элементы крупного предприятия на основании учета технических возможностей и портфеля заказов требуется определить структуру продукции, обеспечивающую максимальную массу покрытия затрат относительно данного периода. Во второй главе приводится математическая модель проблемы, которая представляет собой задачу по линейному программированию. Поскольку задача из-за ее больших масштабов не может быть решена непосредственно, в статье излагается также пригодный для решения способ декомпозиции.

В 3-ей главе автор рассматривает вопрос перевода этого способа на ЭВМ: излагает соображения, условия и ожидания, которые были приняты во внимание в ходе выбора и программирования алгоритмов, обеспечивающих реализацию отдельных этапов декомпозиции. Занимается также со связью всего этого с формулировкой и, соответственно, переформулировкой математической модели. Дальнейшие следствия подытоживает в 5-ой главе. Здесь имеет место написание задачи, одно решение которой дает упомянутый выше способ и производится исследование связи этой задачи с первоначальной задачей, сформулированной во 2-ой главе.