VIZVÁRI BÉLA

Néhány újabb hazai eredmény a diszkrét programozásban

1. Diszkrét programozási feladatokról általában*

Diszkrét programozási feladat alatt olyan optimalizálási feladatot értünk, ahol a változók, vagy egyesek közülük nem változhatnak egy tartományon belül folytonosan, hanem csak bizonyos (egymástól elkülönült = diszkrét) értékeket vehetnek fel.

A magyar szakkifejezések az angol nyelvű szakirodalomból származnak. Mivel azonban ott sincs mindenben egységes szóhasználat, ezért nálunk sincs. Így a magyarban a diszkrét programozás mellett, mely egyébként a *discrete programming*-ból származik, szokásos még az egészértékű programozás, mely az *integer programming*-ból jön. Ennek egy további származéka az integer programozás, ahol az első szót latinként foghatjuk fel.

Érdemes összehasonlítást tenni a tematikát illetően a diszkrét programozás és a matematikai programozás egyéb ágai között. Amíg a tisztán folytonos problémák esetében élesen megkülönböztetjük a lineáris és nemlineáris programozást, mintegy két külön diszciplínának tekintve a kettőt, addig a diszkrét programozásban nem ez a helyzet. Kétségtelen azonban, hogy mind a mai napig itt is elsőrendű fontosságúak a lineáris feladatok.

Amíg a lineáris programozásban egységesen tudjuk kezelni valamennyi változó típust (nemnegatív, előjel kötetlen, 0 és 1 között változó stb.), addig nem ez a helyzet a diszkrét programozásban. A változóra tett egészértékűségi követelmény három leggyakoribb alakja a következő:

$$x = 0 \text{ vagy } 1;$$

$$0 \leq x \leq d \text{ és egész;}$$

$$0 \leq x \text{ és egész.}$$

Ezen három eset elviekben is különbözik egymástól. Például a harmadik követelmény mellett lineáris feltételek és általános kvadratikus célfüggvény esetén bizonyítható, hogy nem létezik megoldó algoritmus, azaz olyan garantáltan véges eljárás, mely az optimális megoldást szolgáltatná. Ugyanez nyilván nem igaz az első két típusnál, hiszen ott véges sok esetről van csak szó.

Külön fejezetet képeznek az ún. vegyes diszkrét feladatok, ahol a változók egy részére nincs egészértékűségi követelmény. Itt az alkalmazott módszerek elég változatosak és ezek részben el is mossák a határt a diszkrét programozás és a matematikai programozás egyéb ágai között. Példának okáért a *Benders* dekompozíció, mely a lineáris programozásból ismert *Dantzig-Wolfe* dekom-

* A dolgozat az MTA SzTAKI Alkalmazott Matematikai Főosztályán működő Diszkrét Programozási Csoportnak az elméleti téren végzett munkáját foglalja össze.

pozíció duálja, a vegyes változós feladatok megoldását tiszta diszkrét (folytonos változókat nem tartalmazó) és lineáris programozási feladatok sorozatainak megoldására vezeti vissza. Ezért a tiszta diszkrét feladatok hatékony megoldásával közelebb kerülünk a vegyes változós feladatok hatékony megoldásához is.

Minél nagyobb értékeket vehetnek fel a változók, annál inkább elveszti a probléma a diszkrét jellegét, vagyis a megfelelő folytonos feladat optimális megoldásából egyszerű módon (pl. kerekítéssel) származtatott egészértékű megoldás kielégíti egy gyakorlati feladat által támasztott igényeket.

Másfelől, ha a változók felülről korlátosak, akkor a probléma visszavezethető elvben arra az esetre, amikor csak a 0 és az 1 értéket vehetik fel.

Nagyon sok olyan gyakorlati probléma van, amely alternatív döntéseket tartalmaz. Ezeknek diszkrét programozási modellezése esetén a numerikus feladat számos ilyen 0—1 ún. döntési változót fog tartalmazni. (Ugyanis vagy megvalósítunk egy lehetséges alternatívát, vagy nem, közbülső eset nincs, vagy ha volna, az önálló alternatívaként fogható fel.)

Ezek azok az okok, amelyek miatt a diszkrét programozás irodalmában kiemelkedő helyen állnak a csak 0—1 változókat tartalmazó feladatok. Kutatásainkban ugyanezen okok miatt fektettünk mi is igen nagy súlyt az ilyen problémákra.

2. Általános megoldási módszerek

A diszkrét programozás első módszere a híres *Gomory*-eljárás volt. Mindmáig ez az a módszer, amely elméleti szempontból a legtöbbet tudja, nevezetesen az egészértékűségi feltételek az előző fejezetben tárgyalt bármelyik formában megadhatók számára. Azonban a tapasztalat azt mutatja, hogy éppen ezen általánossága miatt (bizonyos ismert kivételektől eltekintve) lassú az eljárás. Ezért hamar megindult a kutatás olyan eljárások irányába, amelyek kevésbé általánosak, de jobban ki tudják használni a probléma speciális tulajdonságait és így hatékonyabbak.

Két jelentős eljáráscsaládot fejlesztettek ki: a korlátozás és szétválasztás típusú módszereket és a leszámlálási eljárásokat. Mindkét módszer elsősorban akkor alkalmazható, ha az összes eseteknek (vagyis az egészértékű változók lehetséges értékkombinációnak) a száma véges. Közös lényegük, hogy végigvizsgálják az összes esetet, de döntő többségüket csak implicit módon. A nemzetközi irodalomban a hetvenes évek közepére rögzült az összefoglaló közös nevük: fakereső eljárás. Az elnevezés onnan ered, hogy az ismétlések elkerülése végett (így biztosítható az eljárás végessége) a megoldások részhalmaiból egy irányított fát építenek fel és ezt járják be, mialatt minden megoldást megvizsgálnak (a többséget persze csak implicit módon).

A korlátozás és szétválasztás módszere eredetileg a nemlineáris programozásban született meg, de nagyon sikeresen alkalmazták ezen a területen is. A ma kapható valamennyi kereskedelmi programcsomag ezen az elven alapul. Ennek az az oka, hogy itt a feladat ún. folytonos relaxációjára támaszkodnak, vagyis elhagyják az egészértékűségi követelményeket, csak az előjelre, illetve a változó nagyságára való korlátokat tartják meg. Lineáris feladat esetén így egy lineáris programozási feladatot kapunk. Valamennyi nagyobb számítógépes cég hatékony LP programcsomaggal rendelkezik, így ez viszonylag kis erőbefektetéssel kiterjeszthető volt a diszkrét programozás irányába.

A leszámlálási algoritmusok ugyancsak széleskörben elérhetők, de kutatók által írt programok formájában. Ennek az az oka, hogy itt a feladat diszkrét tulajdonságai kerülnek előtérbe, nincs feltétlenül szükség hatékony LP-re, amelynek a kifejlesztése önmagában is nagy munka volna.

Természetesen vannak további módszerek is. A teljesség igénye nélkül néhány: dinamikus programozáson alapuló eljárások, az ezekből kifejlődött csoportelméleti módszer, a Gomory-eljárás nyomán kialakult vágás típusú módszerek stb.

Csoportunk elsősorban a leszámlálási módszerekkel foglalkozott. A módszer elméleti alapjainak egy, a korábbi szerzők munkáin alapuló, azokat továbbfejlesztő tárgyalása található [15]-ben, majd ugyanez még részletesebben [17, 2. fej.]-ben.

A feladat úgy fogalmazható meg, hogy adva van egy véges alaphalmaz és ennek egy implicit módon definiált részhalmaza. Ennek az utóbbinak kell az összes elemét előállítani explicit módon. Az eljárás a rendezett pszeudomegoldás fogalmán alapszik. Teljes matematikai pontosságú felépítése az említett művekben található, itt csak röviden ismertetjük. Vezessük be a következő jelöléseket. Legyen $D \subset \mathcal{R}^n$ az alaphalmaz, tehát $|D| < +\infty$. Fel kell tennünk, hogy D elemeit explicit módon ismerjük. Ez nyilván teljesül, ha D az n-dimenziós bináris vektorok halmaza, vagy ütemezési feladatokra gondolva az n elemű permutációk halmaza stb. Legyen $S \subset D$ az implicit módon megadott részhalmaz.

Az algoritmus legfontosabb segédeszközei a tesztek. Általánosan úgy fogalmazhatunk, hogy a teszt egy olyan eljárás, amely egy eldöntendő kérdésre válaszol igennel, vagy nemmel úgy, hogy igen válasz esetén nem téved. Ha például D a bináris vektorok halmaza, akkor egy ilyen jellegzetes kérdés lehet a következő: „Igaz-e, hogy S valamennyi elemében az ötödik komponens értéke 1?" Tehát „igen" válasz esetén az eljárás bizonyította, hogy olyan S-beli vektor, melynek ötödik komponense 0, nem létezik. (Néhány ilyen egyszerű tesztre majd a 4. szakaszban látunk példát.) Az így nyert plusz információt következménynek nevezzük. Ha a példát folytatjuk, akkor most már explicit módon megkövetelhetjük, hogy az ötödik komponens értéke 1 legyen.

Ha a tesztek nem adnak pozitív választ, akkor is megtehetjük, hogy a D halmaznak csak valamely részhalmazát vizsgáljuk. Például csak azokat a vektorokat vizsgáljuk, amelyekben a harmadik komponens értéke 0. Az ilyen értékadást *rögzítés*nek nevezzük, ellentétben a tesztekből származókkal, melyek neve *lekötés*. Azoknak a változóknak a neve, amelyek sem rögzítésben, sem lekötésben nem szerepelnek, szabad változó.

Az értékadások (rögzítések és lekötések) a D alaphalmaznak és ezzel együtt az S halmaznak egy részhalmazát jelölik ki. Ezek a részhalmazok lesznek az említett fa csúcsai. A fa éle egy halmazból, mint csúcsból, a belőle értékadással (rögzítés, lekötés) közvetlenül keletkezett részhalmazhoz vezet.

Az [15]-ben bevezetett új fogalom a rendezett pszeudo-megoldás volt, mely a részhalmazokat az őket kijelölő értékadásokkal és az utóbbiak típusával és sorrendjével összekapcsolta.

Az alábbiakban megadjuk az algoritmus vázát, egy algolszerű leírási formában.

```
1. procedure IMPLICIT-LESZÁMLÁLÁS
2. Begin
3.    ELSŐITER: = true
4.    while ELSŐITER or RÖGZITETT-VÁLTOZÓ do
5.    Begin
         ELSŐITER: = false
```

```
 7.    TESZTELÉS
 8.    if not ÜRES-ÁG rhen
 9.      Begin
10.        if VAN-SZABAD-VÁLTOZÓ then RÖGZITÉS
11.          else
12.            Begin
13.              MEGENGEDETTSÉG-VIZSGÁLAT
14.              ÁGCSERE
15.            end
16.      end
17.   else   ÁGCSERE
18. end
19. end IMPLICIT-LESZÁMLÁLÁS
```

Az egész algoritmus három legfontosabb eljárása a TESZTELÉS, a RÖGZITÉS és az ÁGCSERE. Az utóbbi az előző kettő eredményétől függően akkor lép működésbe, ha a fa valamely ágát teljesen megvizsgáltuk és új ágra (részhalmazra) kell áttérni. Az első kettő felfogható úgy is, hogy azok rekurzív módon hívják egymást, ezzel biztosítva az algoritmus előrehaladását. Ennek alapján dolgozott ki Bíró Miklós egy a fentinél sokkal részletesebb, hatékony algoritmikus keretet, mely igen kevés további munkával adaptálható volt különböző feladatokra is (lineáris 0—1 feladat, polinomiális 0—1 feladat, korlátos változós nem 0—1 feladat).

3. Heurisztikus módszerek

A heurisztika fogalmát szokás egészen tágan értelmezni, úgy, mint valamilyen stratégia, elv követését a megoldás során. Egy szűkebb, de gyakoribb értelmezés a megoldandó feladat valamilyen közelítő megoldását jelenti. Mi is ebben az utóbbi felfogásban fogjuk a kifejezést használni.

3.1. *Általánosított Lagrange szorzók*

Everett volt az, aki 1963-ban javasolta a Lagrange szorzók használatát a matematikai programozás ezen ágaiban. Tegyük fel, hogy a következő feladatot kell megoldanunk:

$$\begin{aligned} &\max f(\mathbf{x}) \\ &g_i(\mathbf{x}) \leq b_i \quad i = 1, \ldots, m, \\ &\mathbf{x} \in S, \end{aligned} \tag{3.1}$$

ahol $\mathbf{x}$ n-dimenziós vektor, S az n-dimenziós euklideszi tér tetszőleges, rögzített részhalmaza és f és g_i $(i = 1, \ldots, n)$ tetszőleges n-változós függvények.

A feladat ilyen felírása mögött az rejlik, hogy az összes feltétel gyakran két csoportra osztható: vannak algebrailag kezelhető feltételek és vannak olyanok, amelyeket azonnal alkalmazni tudunk a változókra. Mindenesetre a diszkrét programozási feladatok ilyenek. Például a lineáris 0—1-es feladat esetén a probléma

$$\begin{aligned} &\max \mathbf{c}^T\mathbf{x} \\ &A\mathbf{x} \leq \mathbf{b} \\ &\mathbf{x} \in \{0, 1\}^n \end{aligned} \tag{3.2}$$

alakú (**c**, illetve **b** megfelelő dimenziós vektor A pedig egy alkalmas mátrix). Itt S a bináris vektorok halmaza.

Everett azt javasolta, hogy a (3.1) feladat helyett foglalkozzunk az alábbival:

$$\max_{\mathbf{x} \in S} \left(f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) \right), \tag{3.3}$$

ahol $\lambda_i (i = 1, \ldots, m)$ rögzített nemnegatív szám. Emögött az áll, hogy a (3.3) feladatban nincsenek a megoldást nehézkessé tevő algebrai feltételek, hanem csak azok, amelyek explicit előírásokat tartalmaznak a változókra. Így a (3.3) feladat megoldása várhatóan sokkal könnyebb, mint a (3.1)-é és ezért várhatóan érdemes a (3.3) alakú feladatok egy sorozatát megoldani a (3.1) közvetlen megoldása helyett. Hogy ez mennyire így van, mutatja, hogy a (3.2)-nek megfelelő (3.3) típusú feladat:

$$\max_{\mathbf{x} \in \{0,1\}^n} (\mathbf{c}^T - \lambda^T A)\mathbf{x} \tag{3.4}$$

Tehát csak a $\mathbf{c}^T - \lambda^T A$ vektor komponenseinek előjelét kell megvizsgálni az optimális megoldás meghatározásához.

Lényegében már *Everett* bizonyította a következőt.

1. *optimalitási kritérium:* Ha valamely rögzített λ-ra a (3.3) feladat optimális megoldása **w** és teljesülnek az alábbiak

$$\text{(i)} \quad \lambda_i > 0 \text{ esetén } g_i(\mathbf{w}) = b_i$$
$$\text{(ii)} \quad \lambda_i = 0 \text{ esetén } g_i(\mathbf{w}) \leq b_i$$

akkor **w** optimális megoldása a (3.1) feladatnak.

Az (*i*) és (*ii*) feltételek egyebek mellett azt jelentik, hogy a **w** vektor megengedett a (3.1) feladatra nézvést. Fordítva azonban ez nem igaz, azaz ha egy vektor a (3.3) feladat optimális megoldása és megengedett (3.1)-ben, akkor nem feltétlenül elégíti ki a (*i*) és (*ii*) feltételeket és vonatkozik ez magára a (3.1) feladat optimális megoldására is. Így előfordulhat, hogy már megkaptuk a keresett megoldást, de ezt a tényt nem tudtuk érzékelni.

Ezen a problémán segít a [21] dolgozatban publikált optimalitási kritérium, mely mindannyiszor teljesül, valahányszor az 1. optimalitási kritérium.

3.1. *definíció:* Legyen rögzített $\boldsymbol{\lambda}$ mellett a (3.3) feladat optimális megoldása **w**. Tegyük fel, hogy **w** megengedett megoldása a (3.1) feladatnak. Ekkor a **w** vektor optimalitási tartományának nevezzük a

$$H(\mathbf{w}, \boldsymbol{\lambda}) = \{\mathbf{x}: \ \mathbf{x} \in S; \ \boldsymbol{\lambda}^T(\mathbf{g}(\mathbf{w}) - \mathbf{g}(\mathbf{x})) \geq 0\}$$

halmazt.

Jelölje M a (3.1) feladat megengedett megoldásainak halmazát, azaz

$$M = \{\mathbf{x}: \ \mathbf{x} \in S; \ \mathbf{g}(\mathbf{x}) \leq \mathbf{b}\}.$$

2. *optimalitási kritérium:* Tegyük fel, hogy a $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_r$ szorzókhoz a (3.3) feladat optimális megoldási rendre az $\mathbf{x}_1, \ldots, \mathbf{x}_r$ vektorok voltak, melyek egyben megengedett megoldásai a (3.1) feladatnak is. A megfelelő optimalitási tartományok legyenek $H(\mathbf{x}_1, \boldsymbol{\lambda}_1), \ldots, H(\mathbf{x}_r, \boldsymbol{\lambda}_r)$.

Ha

$$\boldsymbol{M} \subset \bigcup_1^r H(\mathbf{x}_k, \boldsymbol{\lambda}_k), \tag{3.5}$$

akkor

$$\max \{f(\mathbf{x}): \ \mathbf{x} \in \boldsymbol{M}\} = \max \{f(\mathbf{x}_k): \ 1 \leq k \leq r\}.$$

Könnyen konstruálható olyan kisméretű példa, ahol az 1. optimalitási kritérium nem működik, míg a most megadott igen. A (3.5) feltétel ellenőrzése a gyakorlatban megvalósítható, ugyanis érdemes a következő, vele ekvivalens feltételt ellenőrizni

$$\bigcap_1^r \left(S \setminus H(\mathbf{x}_k, \boldsymbol{\lambda}_k)\right) \cap \boldsymbol{M} = \varnothing.$$

Ez pedig nem jelent mást, mint az eredeti feltételekhez r darab

$$\boldsymbol{\lambda}_k^T\big(\mathbf{g}(\mathbf{x}_k) - \mathbf{g}(\mathbf{x})\big) < 0 \tag{3.6}$$

alakú feltétel hozzávételét.

Fontos tulajdonsága a (3.6) alakú feltételeknek, hogy azok a (3.1) feladat eredeti feltételeiből algebrai úton nem származtathatók. Ez a tulajdonságuk lehetőséget ad arra, hogy a Lagrange szorzós eljárást egy leszámlálási algoritmusba ágyazva a leszámlálási fában ugrásokat hajthassunk végre. A módszerrel végzett tapasztalatok azt mutatják, hogy ez a fajta heurisztika alkalmas jó megengedett megoldások generálására.

3.2. *Szomszédság fogalmon alapuló heurisztika*

Tegyük fel, hogy egy véges M halmaz egy kitüntetett pontját keressük. A most ismertetésre kerülő heurisztikus eljárás két eszközt használ fel:

(a) adott egy h függvény az M halmazon, mely azt méri, hogy az M pontjai menynyire közelítik meg a keresett pont tulajdonságait; pontosabban szólva ha

$$x, y \in M \ \text{ és } \ h(x) < h(y),$$

akkor azt mondjuk, hogy az y pont jobb az x pontnál,

(b) minden $x \in M$ esetén adva van M-nek egy $S(x)$ részhalmaza, melybe eső pontokat x szomszédainak nevezzük, úgy, hogy

$$|S(x)| \ll |M|,$$

azaz $S(x)$ sokkal kevesebb pontot tartalmaz M-nél.

Most már maga a módszer egyszerűen megfogalmazható:

```
1.  Begin
2.  k: = 0
3.  x_0: = M-beli pont
4.  G: = {x: x ∈ S(x_k); h(x) > h(x_k)}
5.  while G ≠ ∅ do
6.      Begin
```

7. $k := k + 1$
8. $x_k :=$ G-beli pont
9. $G := \{x: x \in S(x_k);\ h(x) > h(x_k)\}$
10. end
11. end

Tehát az eljárás lényege, hogy egy pontból kiindulva az éppen vizsgált pontot valamelyik nála jobb szomszédjára cseréljük ki. Maga a módszer több helyen felmerül az irodalomban, sőt néhány speciális feladatra vonatkozó egzakt eljárás is ebben a keretben dolgozik.

A (3.2) feladatra úgy alkalmaztuk a módszert, hogy a g függvény a következő volt:

$$h(\mathbf{x}) = \begin{cases} \mathbf{c}^T\mathbf{x}, \text{ ha } \mathbf{x} \text{ megengedett (3.2)-ben} \\ \sum\limits_i \left| b_i - \sum\limits_j a_{ij}\mathbf{x}_j \right|_- \text{ különben,} \end{cases}$$

ahol $|a|_-$ az a szám negatív részét jelöli, azaz

$$|a|_- = \begin{cases} a, \text{ ha } a < 0 \\ 0 \text{ különben.} \end{cases}$$

Az általánosság megszorítása nélkül feltehető, hogy a $\mathbf{c}$ vektor komponensei nemnegatívok, ezért az így megadott függvény bármely megengedett megoldást jobbnak minősít bármely nem-megengedett megoldásnál.

Az eljárás másik lényeges pontja a szomszédok megválasztása. Két különböző szomszédsági rendszert próbáltunk ki: $\mathbf{x}$ szomszédainak tekintettük azokat a pontokat, amelyek pontosan 1, illetve azokat, amelyek pontosan 2 komponensben különböztek tőle. Ha a változók száma n, akkor az így definiált szomszédok száma

$$n, \text{ illetve } \frac{n(n-1)}{2}.$$

Mindkét szám lényegesen kisebb a bináris vektorok 2^n számánál. Mégis a két komponensben eltérőek száma túl nagynak bizonyult, ami abban nyilvánult meg, hogy az algoritmus 8. és 9. sorában megadott lépések végrehajtásához igen sok idő kellett.

A másik variációban gépidő szempontjából is hatékonynak bizonyult a módszer. Három feladatcsoporton próbáltuk ki, ahol az egyes csoportokban a változók és a feltételek száma 50 és 10, illetve 100 és 10, illetve 100 és 30 volt, az egyes csoportba tartozó feladatok száma pedig rendre 10, 10 és 6. Valamennyi feladatra kaptunk megengedett megoldást, általában több, egyre javuló pontokat.

4. Speciális feladatok

A diszkrét programozásban a speciális feladatoknak kettős a szerepe. Egyfelől léteznek önálló alkalmazásaik, másfelől felmerülnek mint részfeladatok a legnehezebb feladatok megoldása során. A később tárgyalandó speciális feladatok elméleti szempontból ugyanolyan nehezek, mint az általános feladat, a gyakorlatban azonban már jelentős különbségek vannak. A hátizsák feladat

esetében akár 20 000 változós feladatokat is meg tudunk oldani, korlátot gyakorlatilag csak a számítógép memóriája szab. Nagyon hasonló a helyzet a halmazfedési feladat esetén is, ahol rövid CPU idővel lehet megoldani néhány száz változót és feltételt tartalmazó feladatokat is. Az ún. többfeltételes hátizsák feladat esetén a törekvés mindenütt jó megengedett megoldások generálása, de ebben az értelemben szintén tudunk kezelni többszáz változós problémákat. Az itt említett adatok egybevágnak a nemzetközi irodalomban közölt tapasztalatokkal.

Néhány példa a speciális feladatok önálló alkalmazásaira: kiértékelt kutatási pályázatok között a kutatásra fordítható anyagi kereteket optimálisan szétosztani a hátizsák feladat segítségével lehet. A hátizsák feladat egy további alosztálya, ahol a változók ún. általánosított felső korlátozás alá esnek, alkalmas optimális technológia választására, ha egy adott munkadarab-halmaz esetében minden munkadarabra véges sok technológia van megadva, és korlátozott a teljes megmunkálási idő. Optimális egységcsomagrendszer kialakítására vagy légi járatoknál a személyzet beosztására alkalmazható a halmazfedési feladat. A többfeltételes hátizsák feladat lehet a matematikai modellje darabolási és egyéb térkitöltési problémáknak vagy termékösszetétel meghatározásának, ha kissorozatú termelés folyik.

4.1. *A hátizsák feladat*

Most a

$$\begin{aligned} &\max \sum_{j=1}^{n} c_j x_j \\ &\sum_{j=1}^{n} a_j x_j \leq b \\ &x_j \in \{0, 1\} \quad j = 1, \ldots, n \end{aligned} \tag{4.1}$$

feladatot vizsgáljuk, ahol tehát egyetlen lineáris feltétel van, az együtthatók pedig tetszőleges egészek.

A feladatnak a nemzetközi irodalomban általánosan elterjedt neve onnan ered, hogy ez modellezi a következő döntési problémát. Egy turista a hátizsákjában egy bizonyos súlynál (b) nem akar többet cipelni. Adott minden tárgy súlya (a_j) és eszmei értéke, amekkora hasznot hoz a kiránduláson. Itt x_j döntési változó, értéke 1 ha j. tárgyat elvisszük, különben 0. A (4.1) feladat megoldásával megkapjuk, hogy mi az elviendő tárgyak optimális halmaza.

Nagyon könnyen látható, hogy a (4.1) feladat azonnal visszavezethető egy olyanra, ahol valamennyi együttható pozitív. Így a továbbiakban ezt mi is feltesszük.

Diszkrét programozásban szokás hátizsák v. hátizsák típusú feladatról beszélni, ha csak egyetlen (nem feltétlenül lineáris) feltételünk van, illetve többfeltételes hátizsák feladatról akkor, ha egy $A\mathbf{x} \leq \mathbf{b}$ feltételekkel adott problémában $A \geq 0$. Az utóbbi és a hátizsák feladat is rendelkezik ugyanis azzal a fontos tulajdonsággal, hogy ha egy vektor megengedett, akkor valamennyi, vele összehasonlítható, nem nagyobb vektor is az.

A (4.1) feladat megoldására számos módszer ismeretes. Így [17]-ben is találunk dinamikus programozásra épülő, valamint leszámlálási és korlátozás-

szétválasztási módszert. Ezek az algoritmusok azonban még nem tennék lehetővé a már említett nagyméretű feladatok megoldását. Ehhez az ún. változó redukciós eljárások kidolgozására volt szükség.

A hátizsák feladat egy igen lényeges dologban különbözik az általános diszkrét programozási feladattól. Ebben az esetben ugyanis igen egyszerű megengedett megoldást, sőt jó megengedett megoldást találni, míg az általános esetben ez ekvivalens nehézségű az egész probléma megoldásával.

Tehát valamennyi változó redukciós eljárás abból indul ki, hogy már ismerünk egy megengedett megoldást z_0 célfüggvényértékkel, és csak az ennél jobb pontok érdekelnek bennünket. A módszerek lényege abban áll, hogy megmutatják, hogy az utóbbi követelmény mellett bizonyos változók értéke csak 0 vagy 1 lehet.

A korábban ismert megoldási módszerekben nagy szerepet játszott az a feltételezés, hogy a változóknak egy olyan sorrendjét ismerjük, amelyre igaz az alábbi egyenlőtlenség

$$\frac{c_j}{a_j} \geq \frac{c_{j+1}}{a_{j+1}}, \quad j = 1, \ldots, n-1. \tag{4.2}$$

Ezen sorrend ismeretében lehet megoldani a hátizsák feladat folytonos változatát, vagyis azt a feladatot, amely a (4.1) feladatból úgy keletkezik, hogy a speciális egészértékűségi követelményt a nála gyengébb

$$0 \leq x_j \leq 1 \quad j = 1, \ldots, n$$

feltétellel helyettesítjük. Igaz ugyanis a következő állítás.

4.1. *tétel:* Ha a (4.1) feladat változói a (4.2) sorrendben vannak, akkor a megfelelő folytonos feladat $\bar{\mathbf{x}}$ optimális megoldása a következő alakú: létezik egy

$$p \text{ index } (1 \leq p \leq n), \text{ hogy}$$
$$\bar{x}_1 = \ldots = \bar{x}_{p-1} = 1$$

és

$$\bar{x}_{p+1} = \ldots = \bar{x}_n = 0.$$

A p indexet pivot indexnek nevezik, az x_p az egyetlen változó, mely a (4.1) feladat folytonos változatának optimális megoldásában tört értéket vehet fel. További fontos szerepet játszik a

$$q_p = \frac{c_p}{a_p} \tag{4.3}$$

hányados.

4.2. *tétel:* Legyen $q \geq 0$ tetszőleges valós szám és z_0 a (4.1) feladat egy megengedett megoldásának célfüggvényértéke, továbbá

$$M(q) = \max \{\mathbf{c}^T\mathbf{x} + q(b - \mathbf{a}^T\mathbf{x}): \ \mathbf{x} \in \{0,1\}^n\}.$$

Ekkor ha

$$M(q) - |c_j - qa_j| < z_0,$$

akkor a (4.1) feladat x^* optimális megoldásában

$$c_j - qa_j < 0 \text{ esetén } x_j^* = 0$$

és

$$c_j - qa_j > 0 \text{ esetén } x_j^* = 1.$$

Ennek a tételnek a segítségével azok a változók, melyekre a (4.4) feltétel teljesül eliminálhatók és így a ténylegesen megoldandó feladat mérete redukálódik. Bár a tétel tetszőleges q-ra igaz, a gyakorlatban azonban itt a q_p értéket célszerű alkalmazni, $M(q)$ értéke ugyanis itt a legkisebb.

A rendezésnek itt nemcsak a q_p érték meghatározásánál van szerepe, hanem a z_0-t is ennek a segítségével szokás megkapni. A mohó eljárás, mely a legtermészetesebben alkalmazható heurisztikus eljárás szintén felhasználja a (4.2) sorrendet.

A rendezésből származó mindkét információt megkaphatjuk a rendezés nélkül is. A módszer ilyen gyorsítási lehetőségét tárgyalja az [5] dolgozat. Vizsgáljuk először a célfüggvény becslésének kérdését. Tulajdonképpen az optimális célfüggvényértékre bármilyen (nem feltétlenül alsó) becslést alkalmazhatunk. Ha a becslés indokolatlanul magas volt, akkor a redukált feladatnak vagy nem lesz megengedett megoldása, vagy a redukált feladat optimális célfüggvényértéke a becslés alá esik. Ha azonban a redukált feladat optimális célfüggvényértéke a becslés fölé esik, akkor ez a tény mindenképpen igazolta a becslés jogosságát. A becsült optimum módszer lényege tehát abból áll, hogy egy egyszerűen számítható becslésből kiindulva redukáljuk a feladatot. Szükség esetén a becslést addig csökkentjük, míg az a redukált feladaton is jogosnak nem bizonyul.

Lényegében ugyanez a gondolat felhasználható az optimum értékének és a hozzá tartozó szorzónak az együttes megkeresésére. Ha ismerünk egy olyan intervallumot, amibe az optimális szorzó belesik, akkor ezt az intervallumot egyre szűkíthetjük. Ha $[q_1, q_2]$ ez az intervallum, a következő pedig $[q_1', q_2']$, akkor vagy $q_1 = q_1'$ vagy $q_2 = q_2'$ és q_2' (illetve q_1') $\in (q_1, q_2)$. A becsült optimum módszer alkalmazása után a redukált (kisméretű) feladat megfelelő (4.3) értéke adja q_2'-t (illetve q_1'-t).

Ezeknek a módszereknek a számítástechnikai jelentőségét mutatja, hogy a hagyományos redukciós módszerekhez viszonyítva a szükséges *CPU* idő 750, illetve 1000 változó esetén 13, illetve 10%-ra esett vissza.

4.2. *A halmazfedési feladat*

A halmazfedési feladat szintén egy speciális alosztálya a 0—1-es problémáknak, itt azonban a feltételek száma tetszőleges lehet, míg az együtthatók értéke — akárcsak a változóké — csak 0 vagy 1. Pontosabban szólva a következő feladatról van szó:

$$\begin{aligned} &\min \sum_{j=1}^{n} c_j x_j \\ &\sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad i = 1, \ldots, m \\ &x_j \in \{0, 1\}, \quad j = 1, \ldots, n, \end{aligned} \tag{4.5}$$

ahol c_j és b_i tetszőleges pozitív egészek, míg a_{ij} értéke vagy 0 vagy 1.

(Könnyen látható, hogy az a_{ij} együtthatókra tett feltevés mellett nem jelenti az általánosság megszorítását, hogy a többi együtthatót pozitívnak tekintjük.) Mint triviális esetet ugyancsak kizárjuk a további vizsgálódásból azt az esetet, amikor valamely x_j változó valamennyi a_{ij} együtthatója 0.

A (4.5) feladat a következőképpen interpretálható. Az egyenlőtlenségek bal oldalán álló együtthatókból alkotott mátrix legyen A. Jelölje ennek j-edik oszlopát $\mathbf{a}_j$. Az $\mathbf{a}_j$ vektorok egy véges, pontosan m elemű halmaz bizonyos részhalmazai karakterisztikus vektoraiként foghatók fel. A feladat ezen részhalmazok közül kiválasztani azokat, amelyek együttesen minimális összsúllyal fedik le az alaphalmaz minden elemét előre adott (b_i) vagy annál nagyobb multiplicitással.

Külön szokás vizsgálni a (4.5) feladatnak azt a további részosztályát, amikor valamennyi b_i értéke 1. Ha ezt a megszorítást nem tesszük fel, akkor a feladat ekvivalens azzal, ahol a célfüggvényben maximalizálás van minimalizálás helyett és a feltételek „kisebb-egyenlő" alakban adottak a „nagyobb-egyenlő" helyett, az együtthatókra pedig a fenti kikötések állnak. Az irodalomban az ilyen alakú problémákat halmaz kitöltési feladatnak nevezik. Végül ezen feladatcsalád utolsó tagja az, amikor a feltételekben pontos egyenlőséget követelünk meg; neve halmaz felbontási feladat. Az utóbbi kettővel most nem foglalkozunk.

A halmazfedési feladatnak számos alkalmazása ismeretes [17] az ütemezéselméletben, műszaki, valamint speciális közlekedési problémák megoldásában.

Mivel a most vizsgált probléma is rendelkezik a hátizsák feladatnak azzal a tulajdonságával, hogy egyszerű módon lehet (jó) megengedett megoldást találni, ezért az elméleti vizsgálatok szempontjából érdekessé váltak az optimalitási kritériumok.

4.1. *definíció:* A (4.5) feladat egy $\mathbf{x}_0$ megengedett megoldását minimálisnak nevezzük, ha

$$\forall \mathbf{x} \in \{0, 1\}^n,\ \mathbf{x} \leq \mathbf{x}_0,\ \mathbf{x} \neq \mathbf{x}_0 \text{ esetén } A\mathbf{x} \not\geq \mathbf{b}.$$

Nyilvánvaló a célfüggvény pozitivitása miatt, hogy optimális megoldás csak minimális lehet.

Először azt az esetet fogjuk vizsgálni, amikor a jobb oldal azonosan 1.

4.3. *tétel* [1]: Legyen $\bar{\mathbf{x}}$ a (4.1) feladat egy tetszőleges minimális megoldása. Legyen továbbá

$$J = \{j\colon \bar{x}_j = 1\}$$

$$H_j = \{i\colon a_{ij} = 1;\ \forall k \in J \setminus \{j\} \text{ esetén } a_{ik} = 0\},\ j \in J$$

$$\bar{r}_j = \frac{c_j}{|H_j|} \qquad j \in J$$

$$r_j^0 = \frac{c_j}{\sum_{i=1}^{m} a_{ij}}. \qquad j = 1, \ldots, n$$

Ha $\forall j \in J$ és $\forall k \in \{1, \ldots, n\} \setminus J$ esetén

$$\bar{r}_j \leq r_k^0,$$

akkor $\bar{\mathbf{x}}$ a (4.1) feladat optimális megoldása.

(Megjegyezzük, hogy $\bar{\mathbf{x}}$ minimalitásából következik, hogy $|H_j| \geq 1$.)
Másfelől az imént bevezetett r_j^0 számok segítségével a célfüggvény alulról megbecsülhető.

4.4. *tétel* [17]: Legyenek az r_j^0 $(j = 1, \ldots, n)$ számok ugyanazok mint fent, továbbá

$$f_i = \min \{r_j^0:\ 1 \leq j \leq n;\ a_{ij} = 1\}.$$

Ekkor a (4.1) feladat z^* optimális célfüggvényértékére igaz, hogy

$$z^* \geq \sum_{i=1}^{n} f_i.$$

Mindezek alapján már el lehetett készíteni egy korlátozás és szétválasztás típusú eljárást, amellyel 100 változóig sikerült feladatokat megoldani.
Az alábbiakban ismertetendő eredmények [19] már az általános esetre vonatkoznak, tehát a b_i együtthatók tetszőleges pozitív egészek lehetnek.
Mint eddig is $\bar{\mathbf{x}}$ a (4.1) feladat egy minimális megengedett megoldását, J pedig $\bar{\mathbf{x}}$ azon komponensei indexeinek halmazát jelöli, melyek értéke 1. Legyen továbbá I azon feltételeknek a halmaza, amelyeket $\bar{\mathbf{x}}$ egyenlőséggel teljesít, azaz ha $\mathbf{A}_i$ $(i = 1, \ldots, m)$ jelöli az $\mathbf{A}$ mátrix i-edik sorát, akkor

$$i \in I \Rightarrow \mathbf{A}_i \bar{\mathbf{x}} = b_i \text{ és } i \notin I \Rightarrow \mathbf{A}_i \bar{\mathbf{x}} > b_i.$$

4.2. *definíció:* $\mathbf{A}\ \mathbf{r} \in \mathcal{R}^m$ vektort az $\bar{\mathbf{x}}$ ponthoz tartozó árvektornak nevezzük, ha

(*i*) $\forall j \in J$ esetén $\mathbf{r}^T \mathbf{a}_j \geq c_j$
(*ii*) $\forall i \notin I$ esetén $r_i = 0$.

A 4.3. tétel jelöléseit használva könnyen látható, hogy ha $j_1, j_2 \in J$ és $j_1 \neq j_2$, akkor

$$H_{j_1} \cap H_{j_2} = \emptyset.$$

Így értelmes a következő definíció

$$r_i = \begin{cases} \bar{r}_j, & \text{ha } \exists j:\ i \in H_j \\ 0 & \text{különben.} \end{cases}$$

Az így meghatározott vektor az előbbi definíció értelmében árvektor lesz.

4.5. *tétel:* Legyen $\mathbf{r}$ az $\bar{\mathbf{x}}$ ponthoz tartozó árvektor, $\mathbf{v} \in \mathcal{R}^n$ vektor pedig a következő

$$v_j = \begin{cases} 1, & \text{ha } j \notin J \text{ és } \mathbf{r}^T \mathbf{a}_j - c_j > 0 \\ 0 & \text{különben.} \end{cases}$$

Ha $\mathbf{x}$ a (4.1) feladat egy tetszőleges olyan megengedett megoldása, amelyre

$$\mathbf{c}^T \mathbf{x} < \mathbf{c}^T \bar{\mathbf{x}},$$

akkor $\mathbf{x}$ kielégíti az alábbi feltételt

$$\mathbf{v}^T \mathbf{x} \geq 1. \tag{4.6}$$

Megjegyezzük, hogy a $\mathbf{v}$ vektor definíciójából azonnal látszik, hogy

$$\mathbf{v}^T\bar{\mathbf{x}} = 0,$$

tehát a (4.6) egyenlőtlenség levágja az $\bar{\mathbf{x}}$ pontot. E tény jelentőségét az adja meg, hogy (4.6) hozzávétele a feltételekhez nem rontja el a feladat típusát.

Ez a tétel közös általánosítása *Belmore—Ratliff* és *Balas* eredményeinek és a 4.3 tételnek. Az utóbbi esetben a fent megadott árvektor esetén $\mathbf{v} = \mathbf{0}$ adódik. Ekkor azonban az egyenlőtlenség kielégíthetetlen, ami az $\bar{\mathbf{x}}$ pont optimalitását bizonyítja.

A következő tételben az y valós szám pozitív részét $|y|_+$ jelöli

4.6. *tétel.* Legyen $\mathbf{w} \in \mathcal{R}^m$ tetszőleges vektor, z természetes egész és

$$t(w) = \mathbf{w}^T\mathbf{b} - \sum_{j=1}^{n} |\mathbf{w}^T\mathbf{a}_j - c_j|_+.$$

Ekkor:

(*i*) A (4.5) feladat minden $\mathbf{x}$ megengedett megoldására

$$\mathbf{c}^T\mathbf{x} \geq t(\mathbf{w}).$$

(*ii*) Ha $\mathbf{x}$ a (4.5) feladat megengedett megoldása és $\mathbf{c}^T\mathbf{x} < z$, akkor

$$c_j - \mathbf{w}^T\mathbf{a}_j \geq z - t(\mathbf{w}).$$

esetén

$$x_j = 0.$$

(*iii*) Az alábbi két állítás közül pontosan az egyik igaz:

(a)Ha az $\mathbf{x}$ vektor olyan, hogy

$$x_j = \begin{cases} 1, & \text{ha } \mathbf{w}^T\mathbf{a}_j > c_j \\ 0 & \text{különben,} \end{cases}$$

akkor $\mathbf{x}$ megengedett megoldás.

(b) Létezik egy $i \in \{1, \ldots, m\}$ index és egy $\delta > 0$ szám, hogy

$$\bar{\mathbf{w}}^T = (w_1, \ldots, w_{i-1}, w_i + \delta, w_{i+1}, \ldots, w_n)$$

esetén

$$t(\bar{\mathbf{w}}) > t(\mathbf{w}).$$

A tétel egyes részállításai jól felhasználhatók egy megoldó algoritmus során. Hiszen (*i*) alsó korlátot ad az optimális célfüggvényértékre, mely (*iii*) alapján esetleg tovább javítható. Ugyancsak (*iii*) alapján új megengedett megoldások generálhatók. Végül (*ii*) segítségével a feladat mérete redukálható.

Ezen módszerekkel sikerült megoldani 2%-os sűrűség esetén 500 feltételt és 3000 változót tartalmazó feladatokat is. Nagyobb sűrűség esetén pedig a problémákat legfeljebb 50 változós, immár nehéz, problémákká sikerült redukálni.

4.3. *Az általánosított hátizsák feladat*

Az irodalomban külön szokás vizsgálni azt a feladatot, amikor valamennyi együttható nemnegatív. Erre elsőként *Toyoda* publikált hatékony heurisztikus eljárást. Az ő célja egyetlen jó közelítő pont megadása volt. Az általunk kifejlesztett, ugyancsak heurisztikus eljárás esetében azonban arra törekedtünk, hogy minél több jó megengedett megoldáshoz jussunk. Számítógépes kísérleteket 30 feltételt és 200 változót tartalmazó feladatokig végeztünk. Néhány szekundum CPU idő felhasználásával számos igen jó ponthoz jutottunk. Úgy tűnik, hogy korlátot itt is elsősorban a memória nagysága szab.

5. Dualitás

A dualitás fogalma a diszkrét programozásban nem olyan jól kidolgozott, szép elmélet, amint a lineáris programozásban megszoktuk. Ennek az az oka, hogy míg az utóbbi esetben a duális feladat ugyanolyan természetű mint a primál feladat, sőt ez a duál feladat duálja, addig a diszkrét programozásban ugyanannak a problémának több duálja is értelmezhető és ezek általában folytonos feladatok.

Most röviden megmutatjuk, hogy a duál feladatot milyen általános keretben szokás értelmezni. A primál feladat

$$\max f(\mathbf{x}) \tag{5.1}$$
$$\mathbf{x} \in M,$$

ahol M valamely megszámlálható halmaz, $f(\mathbf{x})$ M-en értelmezett tetszőleges függvény. Bevezetve új, duál változókat (a továbbiakban s-sel jelöljük őket), egy új duál $d(\mathbf{s}, \mathbf{x})$ célfüggvényt írunk elő, mely valamilyen egyedi (a dualitás típusától függő) módon van kapcsolatban az (5.1) feladattal. A duál változók értéküket csak az előre adott Q halmazból vehetik fel. Végül egy P_s halmazt választunk, mely függ a duál változóktól és teljesül rá,hogy

$$M \subset P_s. \tag{5.2}$$

A $d(\mathbf{s}, \mathbf{x})$ függvény következő nyeregpontját keressük:

$$\min_{s \in Q} \max_{x \in P_s} d(\mathbf{s}, \mathbf{x}). \tag{5.3}$$

Itt (5.2) miatt a rögzített s mellett fellépő belső feladat vagy (5.1)-nek egy relaxációja, vagy egy nagyon egyszerűen kezelhető feladat.

Általában nem garantálható az, amit a lineáris programozás dualitás tétele biztosít, nevezetesen, hogy az optimális célfüggvényértékek egyenlők. Ha z, illetve w az (5.1), illetve (5.3) feladat optimuma, akkor többnyire csak

$$z \leq w$$

igaz. Ez a tulajdonság felveti algoritmikus szempontból a dualitás [pontosabban szólva valamely (5.3) feladat egzakt megoldása] alkalmazhatóságának kérdését. Ugyanis $z < w$ esetben (5.3) optimumhelyének **x** része nem lesz M-beli. Ez az oka, hogy a Lagrange szorzókat, amelyek nagyon jól felhasználhatók megengedett megoldások generálására, a 3. szakaszban tárgyaltuk és nem itt.

5.1. *Az s-feltételek*

Az egyik leggyakrabban használt relaxáció az, amikor a feltételeket egy következményükkel helyettesítjük.
Tekintsük tehát az

$$M = \{\mathbf{x}:\ \mathbf{x} \in \{0,1\}^n,\ A\mathbf{x} \leq \mathbf{b}\}$$

halmazt, ahol A, illetve $\mathbf{b}$ egy $m \times n$ mátrix, illetve m dimenziós vektor. Az M halmaz definíciójában szereplő egyenlőtlenségek egy következményét úgy kapjuk meg, hogy nemnegatív súlyokkal összegezzük őket. Tehát a fenti jelöléseket használva

$$Q = \mathcal{R}_+^m,\ P_s = \{\mathbf{x}:\ \mathbf{x} \in \{0,1\}^n;\ \mathbf{s}^T A\mathbf{x} \leq \mathbf{s}^T\mathbf{b}\}.$$

A megoldandó primál feladat pedig

$$\max \mathbf{c}^T\mathbf{x}$$
$$\mathbf{x} \in M;$$

ahol $\mathbf{c}$ rögzített vektor.

A legtöbbet vizsgált $d(\mathbf{s}, \mathbf{x})$ függvény itt maga a célfüggvény, azaz

$$d_1(\mathbf{s}, \mathbf{x}) = \mathbf{c}^T\mathbf{x}.$$

Tehát ebben az esetben (5.3) megoldásával a lehető legjobb felső becslést akarjuk kapni. *Geoffrion* vezette be a következő duál függvényt.

$$d_2(\mathbf{s}, \mathbf{x}) = (\mathbf{c}^T - \mathbf{s}^T A)\mathbf{x} + \mathbf{s}^T\mathbf{b} - z,$$

ahol a z konstans a

$$-\mathbf{c}^T\mathbf{x} \leq -z$$

célfüggvény-feltétel jobb oldala. A d_2 függvény tulajdonképpen a Lagrange-szorzók alapján is származtatható volna. A duál célfüggvény ilyen választása valamilyen értelemben azt célozza, hogy a P_s halmaz számossága kicsi legyen, azaz P_s a lehető legkevesebb nem M-beli pontot tartalmazza.

Az alább ismertetendő új duálfeladat az előbbiek közös általánosításának tekinthető, annak ellenére, hogy első látásra úgy tűnik, mintha itt nem nyeregpontot keresnénk. Valamennyi eredmény a [7] dolgozatból való.

Legyen $\xi \in \{0,1\}^n$ páronként független valószínűségi változókból álló vektor és

$$P(\xi_j = 1) = 1 - P(\xi_j = 0) = p_j.$$

Ekkor annak a valószínűségét akarjuk minimalizálni, hogy egy P_s-beli pont nem eleme M-nek. Ezt pedig úgy lehet elérni, ha $\mathbf{s}$-t úgy választjuk meg, hogy P_s komplementerének a mértéke a lehető legnagyobb legyen. Így az új duálfeladat

$$\max P(\mathbf{s}^T A\xi > \mathbf{s}^T\mathbf{b}) \tag{5.4}$$
$$\mathbf{s} \geq \mathbf{0}$$

lesz.

Természetesen nem közvetlenül az (5.4) feladatot oldjuk meg, mert a benne szereplő valószínűségi mérték nehezen kezelhető. Az igazán érdekes esetek

azok, ahol n értéke nagy, akkor azonban közelíthetünk a normális eloszlással. Pontosabban szólva a $\zeta = \mathbf{s}^T A\xi$ valószínűségi változóra igaz, hogy

$$E(\zeta) = \mathbf{s}^T A\mathbf{p}$$
$$\sigma^2(\zeta) = \mathbf{s}^T ADA^T\mathbf{s},$$

ahol az $n \times n$-es D mátrix elemei a következők

$$d_{ij} = \begin{cases} 0, & \text{ha } i \neq j \\ p_j(1 - p_j), & \text{ha } i = j. \end{cases}$$

A ζ-ből kapott normált valószínűségi változó eloszlásfüggvényét jól közelíthetjük az $N(0, 1)$ eloszlás Φ eloszlásfüggvényével. Mivel Φ monoton növő és független $\mathbf{s}$-től, ezért

$$\Phi\left(\frac{-\mathbf{s}^T\mathbf{b} + E(\zeta)}{\sigma(\zeta)}\right)$$

maximalizálása ekvivalens az abszcissza maximalizálásával. Így az (5.4) helyett alkalmazott közelítő feladat a következő

$$\max_{\mathbf{s} \geq \mathbf{0}} \frac{\mathbf{s}^T A\mathbf{p} - \mathbf{s}^T\mathbf{b}}{\sqrt{\mathbf{s}^T ADA^T\mathbf{s}}}. \tag{5.5}$$

Az egész megközelítés értelmét az alábbi tételek fejtik ki

5.1. *tétel:* Tetszőleges $\mathbf{s} \geq \mathbf{0}$ esetén

$$\left| P(\mathbf{s}^T A\xi > \mathbf{s}^T\mathbf{b}) - \Phi\left(\frac{\mathbf{s}^T A\mathbf{p} - \mathbf{s}^T\mathbf{b}}{\sqrt{\mathbf{s}^T ADA^T\mathbf{s}}}\right)\right| \leq \frac{M}{\sqrt{n}},$$

ahol az M konstans csak az $\mathbf{A}$ mátrixtól és a $\mathbf{p}$ valószínűségektől függ.

5.2. *tétel:* Tegyük fel, hogy a $\mathbf{c}$ célfüggvényvektor valamennyi komponense pozitív. Legyen $h \geq 2^n$ valós szám és a p valószínűségek az alábbi módon meghatározottak

$$p_j = \frac{1}{1 + h^{c_j}} \qquad j = 1, \ldots, n$$

Legyen továbbá $\bar{\mathbf{s}}$ ebben az esetben az (5.4) feladat optimális megoldása. Ekkor tetszőleges $\mathbf{s} \geq 0$ esetén

$$\max\{\mathbf{c}^T\mathbf{x}: \ \mathbf{x} \in P_{\bar{s}}\} \leq \max\{\mathbf{c}^T\mathbf{x}: \ \mathbf{x} \in P_s\}.$$

5.3. *tétel:* Tegyük fel, hogy a $\mathbf{c}$ célfüggvényvektor valamennyi komponense pozitív. Legyen $\bar{\mathbf{p}}$ a

$$\max \mathbf{c}^T\mathbf{p}$$
$$A\mathbf{p} \leq \mathbf{b}$$
$$\mathbf{0} \leq \mathbf{p} \leq \mathbf{e}$$

feladat optimális megoldása, ahol **e** a csupa 1-esből álló vektor. Tegyük fel, hogy $\bar{\mathbf{p}} \notin \{0,1\}^n$. Legyen továbbá $\bar{\mathbf{s}}$ az (5.5) feladat megoldása ezen $\mathbf{p} = \bar{\mathbf{p}}$ mellett, $\mathbf{s} \geq \mathbf{0}$ tetszőleges vektor. Ekkor

$$\max \{d_2(\bar{\mathbf{s}}, \mathbf{x}):\ \mathbf{x} \in P_{\bar{s}}\} \leq \max \{d_2(\mathbf{s}, \mathbf{x}):\ \mathbf{x} \in P_s\}.$$

Tehát az 5.1 tétel az (5.4) helyett alkalmazandó (5.5) feladat közelítésének pontosságát adja meg. A következő tétel szerint az (5.4) feladat általánosítása a $d_1(\mathbf{s}, \mathbf{x})$ függvény szerint értelmezett dualitásfogalomnak. Végezetül ugyanez mondható az (5.5) feladatról és a $d_2(\mathbf{s}, \mathbf{x})$ függvény szerint dualitásról. Mint korábban már utaltunk rá, az a kikötés, hogy **c** komponensei pozitívak, csak annyit takar, hogy nincs közöttük zérus. Megjegyezzük még, hogy ha az 5.3 tétel feltételei közül $\bar{\mathbf{p}} \notin \{0,1\}^n$ nem teljesül, akkor mind a primál, mind a duál feladat optimális megoldása azonnal előállítható.

A (5.5) feladat numerikus megoldására több módszer is lehetséges. Ezek közül a gyakorlatban a leghatékonyabbnak egy lineáris komplementaritási problémára való visszavezetés bizonyult. Itt ugyanis a változók száma csak az eredeti diszkrét programozási feladat feltételeinek (és nem változóinak) számától függ. *Geoffrion* a $d_2(\mathbf{s}, \mathbf{x})$ szerinti duális probléma megoldását egy nagyméretű lineáris programozási feladat megoldására vezette vissza. A két módszer összevetése összesen 45 db feladaton történt, ahol a legkisebb 4 feltételt és 10 változót tartalmazott, a legnagyobb pedig 10/90, illetve 15/40-es volt. Futási időben az (5.5) feladat megoldása lényegesen (1—2 nagyságrenddel) jobb volt. Ennek egyik oka nyilván az, hogy megelégedtünk az (5.5) probléma egy közelítő megoldásával. Ennek ellenére a generált s-feltételek erősségében, vagyis az s-feltételekből közvetlenül megkapható következmények számában lényeges eltérés nem volt tapasztalható, sőt az esetek többségében ezek azonosnak is voltak.

5.2. *A szubadditív függvényekre alapozott dualitás*

Az itt csak röviden tárgyalandó dualitási fogalom némileg eltér a korábbiaktól.

5.1. *definíció:* Legyen V egy olyan halmaz, melynek elemein egy művelet $(+)$ értelmezve van, és V erre a műveletre nézve zárt. Ekkor egy $f: V \to \mathcal{R} \cup \{+\infty\}$ függvényt szubadditívnak nevezünk, ha

$$\forall x,\ y \in V \text{ esetén } f(x + y) \leq f(x) + f(y).$$

Tekintsük most az általános diszkrét programozási problémát

$$\begin{aligned} &\min \sum_{j=1}^{n} c_j x_j \\ &\sum_{j=1}^{n} \mathbf{a}_j x_j = \mathbf{b} \\ &x_j \geq 0,\ \text{egész}\ j = 1, \ldots, n, \end{aligned} \tag{5.6}$$

ahol c_j valós szám, $\mathbf{a}_j$ valamint **b** valós, m-dimenziós vektorok. Bevezetjük az

$$F = \left\{\mathbf{d}:\ \exists \mathbf{y} \in \mathcal{R}^n;\ \mathbf{y} \geq \mathbf{0};\ \mathbf{d} = \sum_{j=1}^{n} \mathbf{a}_j y_j\right\}$$

jelölést. Most megfogalmazzuk az (5.6) feladat duálját, ahol a változó az f szubadditív függvény lesz.

$$\begin{aligned} &\max f(\mathbf{b}) \\ &f(\mathbf{a}_j) \leq c_j \quad j = 1, \ldots, n \\ &f(\mathbf{0}) = 0 \\ &\forall x, y \in F \text{ esetén } f(x + y) \leq f(x) + f(y). \end{aligned} \tag{5.7}$$

Ismeretes a következő tétel [1].

5.4. *tétel:* Az (5.6) és az (5.7) feladat optimumértékei egyenlők.

Természetesen az (5.7) feladat megoldása még reménytelenebbnek tűnik, mint (5.6)-é, de felmerül a kérdés, hogy nem lehet-e a szubadditív függvények valamilyen részosztályát egyszerű módon meghatározni és az (5.7) feladatot csak erre a részosztályra megoldani. Ekkor természetesen csak alsó korlátot kapunk (5.6) optimumértékére. A kérdést a [8] dolgozat válaszolta meg pozitívan az egydimenziós esetben, és egyben alkalmazta az eredményeket a 6. fejezetben ismertetendő Frobenius problémára.

Tekintsük az a, q_0, $q_1, \ldots, q_s$ természetes számokat, ahol a és q_0 relatív prím, és

$$\prod_{j=0}^{s} q_j > a.$$

Bármely nemnegatív r egész számra $\alpha_j(r)$ $(j = 1, \ldots, s)$ jelentse r-nek a j-edik számjegyét a $q_1, \ldots, q_s$ által meghatározott általánosított számrendszerben, azaz

$$\begin{aligned} r &= \alpha_1(r) + q_1\alpha_2(r) + \ldots + q_1q_2 \ldots q_s\alpha_{s+1}^{(r)} \\ 0 &\leq \alpha_j(r) < q_j, \text{ egész } j = 1, \ldots, s \\ 0 &\leq \alpha_{s+1}^{(r)}, \text{ egész.} \end{aligned} \tag{5.8}$$

Tekintsük az

$F = \{r\colon \exists(x, y),\ x,\ y \geq 0,\ \text{egész},\ r = ax + q_0y\}$ számokat. Tetszőleges r és F esetén legyen $r_0(r)$ az a legnagyobb egész, amelyhez még van olyan nemnegatív egész $y(r)$, hogy kielégítik az

$$r = ar_0(r) + q_0y(r)$$

egyenletet. Mivel a és q_0 relatív prím, ezért $r_0(r)$ létezik, továbbá $y(r) < a$. Definiáljuk az $r_j(r)$ számokat (5.8) alapján

$$r_j(r) = \alpha_j\big(y(r)\big), \quad j = 1, \ldots, s$$

A keresett szubadditív függvény alakja

$$f(r) = \begin{cases} \sum_{j=0}^{s} u_jr_j(r), & \text{ha } r \in F \\ +\infty, & \text{különben} \end{cases} \tag{5.9}$$

ahol $u_0, \ldots, u_s$ rögzített valós számok. Ezeknek azonban bizonyos feltételeknek eleget kell tenniök, hogy $f(r)$ valóban szubadditív legyen.

5.5. *tétel:* Az (5.9)-ben definiált $f(r)$ függvény akkor és csak akkor szubadditív a nemnegatív egészek halmazán, ha fennállnak az alábbiak:

$$(i)\quad u_{j+1} \leq u_j q_j \quad j = 1, \ldots, s-1$$

$$(ii)\quad \text{Ha } \alpha_j = \alpha_j(a) \quad j = 1, \ldots, s \text{ és}$$

$$\alpha_1 = \ldots = \alpha_{k-1} = 0, \text{ akkor}$$

$$q_0 u_0 \leq (\alpha_k - q_k) u_k + (\alpha_{k+1} + 1 - q_{k+1}) u_{k+1} + \ldots$$

$$\ldots + (\alpha_{s-1} + 1 - q_{s-1}) u_{s-1} + (\alpha_s + 1) u_s.$$

A tétel jelentősége abban áll, hogy a szubadditív függvények egy részosztályát egy lineáris feltételrendszerrel választja ki. Bármely hátizsák feladat esetén a és q_0 nyilván megválasztható úgy, hogy $a_j \in F$ $(j = 1, \ldots, n)$ teljesüljön és így az (5.7) feladat — pontosabban annak a megfelelő megszorítása — lineáris programozási feladattá válik.

Természetesen a lineáris függvények maguk is szubadditívok. Ha azonban erre a részosztályra írjuk fel az (5.7) feladatot, akkor nem kapunk új dualitási fogalmat, mert (5.6) folytonos duáljához jutunk.

6. Nem 0—1 problémák

Bármilyen fontos szerepet játsszanak is a 0—1 feladatok a diszkrét programozáson belül, azért persze a többi feladat vizsgálata sem hanyagolható el. Ismerünk számos alkalmazási lehetőséget, ahol a probléma nem bináris természetű.

Mint már korábban említettük, a diszkrét programozás első módszere az ún. *Gomory*-módszer volt. Mindmáig ez a legáltalánosabb módszer, mert az 1. szakaszban említett valamennyi típusú egészértékűségi követelmény kezelhető vele. Maga a módszer az ún. vágás típusú módszerek családjába tartozik. Ezek úgy működnek, hogy meghatározzuk a lineáris programozási relaxáció optimális megoldását. Ha ez nem rácspont, akkor ezt a csúcsot a poliéderből lemetszük egy alkalmas síkkal úgy, hogy a sík egyetlen rácspontot se vágjon le, majd az eljárást megismételjük az immár kisebb poliéderre. Számos ezen az elven működő módszert fejlesztettek ki, de a hozzájuk fűzött reményeket nem váltották be. Ennek egyik oka az lehet, hogy ezek a módszerek túl általánosak, vagyis egy konkrét feladat esetében nem tudják jól kihasználni annak speciális tulajdonságait. A másik, matematikai okra az alábbi tétel vet fényt [11].

6.1. *tétel:* Tekintsük a következő halmazt:

$$X = \{\mathbf{x}:\ \mathbf{x} \in \mathcal{R}^n,\ B\mathbf{x} \geq \mathbf{b}\},$$

ahol B egy $n \times n$-es mátrix, $\mathbf{b}$ egy n-dimenziós vektor. Legyen adva egy tetszőleges $\varepsilon > 0$ és egy tetszőleges m természetes szám. Ekkor létezik egy $n \times n$ D-es mátrix és n-dimenziós $\mathbf{d}$ vektor úgy, hogy valamennyi elemük racionális, továbbá

$$\|\mathbf{b}^i - \mathbf{d}^i\| < \varepsilon \text{ és } |b_i - d_i| < \varepsilon.$$

(ahol $\mathbf{b}^i$, illetve $\mathbf{d}^i$ a B, illetve $\mathbf{D}$ mátrix i-edik sorát jelöli és $\|\cdot\|$ egy norma $\mathcal{R}^n$-ben) és az

$$Y = \{\mathbf{x}: \mathbf{x} \in \mathcal{R}^n, \mathbf{Dx} \geq \mathbf{d}\}$$

halmazban levő rácspontok konvex burkának legalább m extremális pontja van.

Ez a tétel azt mondja, hogy szerencsétlen esetben igen sok csúcsot kell levágni, míg a kívánt optimális megoldáshoz eljutunk.

A továbbiakban először egy olyan módszerről lesz szó, amely elsősorban korlátos változók esetén alkalmazható sikerrel. Utána pedig egy olyan feladatról szólunk, amely egyrészt ezzel a módszerrel is megoldható, másrészt azonban összekötő kapcsot jelent a diszkrét programozás és a számelmélet között.

6.1. *A dinamikus programozás alkalmazása a diszkrét programozásban*

Bellman nevéhez fűződök a következő optimalitási elv [3]. Ha egy döntési sorozat olyan, hogy az egyes döntések nem változtatják meg a korábbi döntések értékét, akkor egy optimális döntési sorozat bármely részsorozata is optimális.

Ennek az elvnek számos alkalmazása ismert a diszkrét programozásban, de más kombinatorikus feladatok esetén is. Így például erre az elvre épül az irányított gráfokban legrövidebb utat kereső egyik közismert eljárás, a korábban már tárgyalt hátizsák feladat egyik gyors egzakt megoldó algoritmusa stb. Ezeknek a problémáknak az a közös jellemzője, hogy a bennük fellépő változók értékkészlete véges. A most [16] alapján bemutatandó eljárás éppen ebben különbözik a többitől.

Tekintsük a következő, nem véges hátizsák feladatot

$$\begin{aligned} &\min \sum_{j=1}^{n} c_j x_j \\ &\sum_{j=1}^{n} a_j x_j \geq z \\ &x_j \geq \mathbf{0} \text{ és egész,} \end{aligned} \tag{6.1}$$

ahol az a_j, c_j együtthatók ($j = 1, \ldots, n$) és a z jobb oldal pozitív egész. A problémát egyelőre mint a z paramétertől függő feladatot vizsgáljuk. Az optimum értékét ezért $f(z)$-vel jelöljük. Egyszerűen bizonyítható a következő két tétel.

6.1. *tétel:* Létezik olyan K egész szám, hogy $\frac{c_1}{a_1} < \frac{c_j}{a_j}$ $(j \geq 2)$ esetén

$$\forall z \geq K \text{ esetén } f(z) = c_1 + f(z - a_1).$$

6.2. *tétel:* Ha $\bar{\mathbf{x}}$ optimális megoldása a (6.1) feladatnak valamely $\bar{z}$ jobb oldal esetén, akkor az $\mathbf{y}$ egész vektor, melyre

$$\mathbf{0} \leq \mathbf{y} \leq \bar{\mathbf{x}}$$

optimális megoldása (6.1)-nek

$$z = \sum_{j=1}^{n} a_j y_j$$

jobb oldal mellett.

A második tétel biztosítja a Bellman-elv érvényesülését a vizsgált feladaton. A 6.1 tétel pedig arra ad lehetőséget, hogy végtelen sok jobb oldalra megoldjuk a feladatot, hiszen, ha $z > K$ és t olyan pozitív egész, hogy

$$z - (t-1)a_1 \geq K > z - ta_1,$$

akkor

$$f(z) = tc_1 + f(z - ta_1).$$

Tehát, ha $z = 1, \ldots, K-1$ esetén ismerjük az optimum értékeket, akkor ezekből azonnal megkapható bármely további z-re az optimum.

Terjedelmi okok miatt itt nem részletezzük az algoritmust, amely a fenti két tétel segítségével adható meg. Röviden csak annyit jegyzünk meg, hogy felhasznál bizonyos elemeket a fentebb már ismertetett korlátozás és szétválasztás eljárásából, és egyszerre oldja meg a (6.1) feladatot valamennyi jobb oldalra.

Ez az utóbbi jellegzetes közös tulajdonsága az összes dinamikus programozáson alapuló algoritmusnak. Ebből származik a módszer előnye és hátránya is. Az előbbi az, hogy ha egy feladatot többször akarunk megoldani változó jobb oldalak, de változatlan egyéb paraméterek mellett, akkor itt gyakorlatilag a második és az utána következő esetekben csak ki kell olvasni a végeredményt. Ennek érdekében azonban elég nagy táblázatokat vagy tömböket kell tárolni, az adott esetben egy K hosszúságú egész vektort.

A dinamikus programozás alkalmazásait a diszkrét optimalizálásban jól foglal, a össze [17] 4. fejezete, mely az eredeti dolgozatokhoz képest a módszerek több javítását is tartalmazza.

6.2. *A Frobenius-probléma*

Frobenius vetette fel a következő problémát, mely az alábbi könnyen bizonyítható tétel alapján fogalmazható meg.

6.3. *tétel.* Legyenek adva az $a_1, \ldots, a_n$ természetes számok úgy, hogy a legnagyobb közös osztójuk 1. Ekkor létezik olyan g egész, hogy a

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

lineáris kifejezés értékkészlete nemnegatív egész változók mellett a g-nél nagyobb valamennyi egészt tartalmazza.

A Frobenius-probléma a legkisebb olyan g megadása, amelyre a tétel állítása igaz. (Ez a g maga még nem tartozik bele az értékkészletbe.)

A keresett számot $g(a_1, \ldots a_n)$-nal fogjuk jelölni.

Ennek a feladatnak a matematika több fejezetében, így a Markov-láncok, a gráfok és a mátrixok elméletében is van alkalmazása, így érthető, ha számosan vizsgálták a kérdést. Terjedelmi okok miatt nem térhetünk ki a korábbi

eredmények részletes ismertetésére. A téma iránt érdeklődő olvasó jó összefoglalót talál [13]-ban. Itt csak a főbb témaköröket említjük meg: egzakt felső korlátok, speciális feladatok explicit megoldása, egzakt megoldó algoritmusok. Nem voltak ismeretesek viszont alsó korlátok, amelyeknek az alkalmazások szempontjából szintén jelentősége van.

Az ebben a témakörben elért eredmények alapja az eredeti probléma két ekvivalens átfogalmazása.

Az elsőnek a lényege egy parametrikus hátizsák feladatra való átfogalmazás. Az általánosság megszorítása nélkül feltehető, hogy

$$a_1 < a_j. \quad j = 2, \ldots, n \tag{6.2}$$

Ekkor az a_j, $(j \geq 2)$ számok a_1 segítségével a következő módon állíthatók elő

$$a_j = c_j a_1 + d_j, \; d_j < a_1, \quad j = 2, \ldots, n \tag{6.3}$$

ahol c_j és d_j természetes számok. Itt ugyanis feltehető

$$d_j \geq 1, \quad j = 2, \ldots, n \tag{6.4}$$

mert ellenkező esetben a_j elhagyható a problémából a megoldás megváltoztatása nélkül.

6.4. *tétel.* Tegyük fel, hogy fennáll (6.2), a c_j, d_j $(j = 2, \ldots, n)$ természetes számok a (6.3) egyenletből származnak. Legyen k természetes szám, melyre $1 \leq k < a_1$, z_k pedig a következő hátizsák feladat optimumértéke

$$\begin{aligned} z_k = \min & \; a_1 \left(x_1 + \sum_{j=2}^{n} c_j x_j \right) + k \\ & -a_1 x_1 + \sum_{j=2}^{n} d_j x_j = k \\ & x_j \geq 0, \text{ egész.} \quad j = 1, \ldots, n \end{aligned} \tag{6.5}$$

Ekkor a Frobenius-probléma megoldása

$$g(a_1, \ldots, a_n) = \max \{z_k - a_1 : \; 1 \leq k < a_1\}.$$

Ebből származtatható az alábbi speciális feladatosztályra vonatkozó felső korlát [23], valamint az azt követő általános alsó korlátok [24].

6.5. *tétel.* Tegyük fel, hogy az $a_1, \ldots, a_n$ számok sorrendje olyan, hogy

$$a_1 < \ldots < a_n,$$

továbbá a (6.3) által definiált c_j számokra

$$c_j = 1 \quad j = 2, \ldots, n$$

áll fenn, valamint $d_2 = 1$. Ekkor

$$g(a_1, \ldots, a_n) \leq \sum_{j=2}^{n} a_j \frac{d_{j+1} - d_j}{d_j} - a_1,$$

ahol $d_{n+1} = a_1$.

6.6. *tétel.* Legyen az i olyan index, hogy

$$\frac{c_i}{d_i} = \min\left\{\frac{c_j}{d_j}:\ 2 \leq j \leq n\right\}. \tag{6.6}$$

Ekkor

$$g(a_1, \ldots, a_n) \geq \frac{c_i}{d_i} a_1^2 - \frac{c_i}{d_i} a_1 - 1.$$

A következőkben $\{t\}$ jelöli a t valós szám törtrészét.

6.7. *tétel.* Legyen i a (6.6) szerinti index, k pedig egy természetes szám, melyre $1 \leq k < a_1$. Az $f_0, f_1, \ldots, f_n$ és a $v, v_1, \ldots, v_n$ számok legyenek a következő módon definiálva

$$f_0 = \left\{\frac{k}{d_i}\right\}$$

$$f_1 = \begin{cases} 1 - \left\{\dfrac{a_1}{d_i}\right\}, & \text{ha } d_i \text{ nem osztója } a_1\text{-nek} \\ 0 & \text{különben} \end{cases}$$

$$f_j = \left\{\frac{d_j}{d_i}\right\} \quad j = 2, \ldots, n$$

$$v_1 = \begin{cases} \dfrac{a_i}{d_i f_1}, & \text{ha } f_1 > 0 \\ +\infty & \text{különben} \end{cases}$$

$$v_j = \begin{cases} \dfrac{c_j d_i - d_j c_i}{d_i f_j}, & \text{ha } f_j > 0 \\ +\infty & \text{különben} \end{cases}$$

$$v = \min\{v_j:\ 1 \leq j \leq n\}.$$

Ekkor

$$g(a_1, \ldots, a_n) \geq a_1 \frac{k c_i}{d_i} + a_1 \left\{\frac{k}{d_i}\right\} v + k - a_1.$$

Könnyen látható, hogy itt a bal oldal a maximumát oszthatósági feltételektől függően az $a_1 - 1$, $a_1 - 2$, $b = \max\{w:\ w < a_1,\ w \equiv d_i - 1 \pmod{d_i}\}$ számok valamelyikében veszi fel. Megjegyezzük, hogy a 6.6 és 6.7 tételben adott alsó korlát éles abban az értelemben, hogy megadható hozzájuk feladatoknak egy-egy olyan végtelen sorozata, ahol n — az a_j számok száma — minden határon túl nő és valamennyi feladatnál az alsó korlát egybeesik a pontos értékkel. A második alsó korlát a Gomory-módszer alkalmazásával kapható meg. A 6.5 tételben szereplő felső korlát ugyancsak a pontos értéket adja, ha a

$$\frac{d_{j+1} - d_j}{d_j}$$

számok egészek.

Jelölje F a következő halmazt

$$F = \left\{ r\colon\ \exists \mathbf{x} \in \mathcal{R}^n,\ \mathbf{x} \geq \mathbf{0},\ \mathbf{x} \text{ egész},\ r = \sum_{j=1}^{n} a_j x_j \right\}.$$

Ekkor nyilván

$$g(a_1, \ldots, a_n) = \max_{r \notin F} r.$$

Legyen $v(r)$ az alábbi hátizsák feladat optimum értéke

$$\begin{aligned} v(r) &= \min \left(-x_1 + \sum_{j=2}^{n} c_j x_j \right) \\ a_1 x_1 + \sum_{j=2}^{n} d_j x_j &= r \\ x_j \geq 0,\ &\text{egész}, \quad j = 1, \ldots, n \end{aligned} \tag{6.7}$$

ahol a c_j, d_j számok (6.3)-ból származnak. Ha a feltételeket kielégítő $\mathbf{x}$ vektor nem létezik, akkor $v(r)$ értéke $+\infty$.

6.8. *tétel* [8]. Az alábbi két állítás ekvivalens.

$$(i)\ \ r \in F$$
$$(ii)\ \ v(r) \leq 0.$$

Így a Frobenius-probléma korábban említett másik átfogalmazása:

$$g(a_1, \ldots, a_n) = \max \{ r\colon\ v(r) > 0 \}.$$

Ezen átfogalmazás segítségével bizonyítható a következő két tétel.

6.9. *tétel* [8]. Ha a c_j, d_j $(j = 2, \ldots, n)$ számokat (6.3), az i indexet (6.6) szerint határozzuk meg, továbbá a $d_2, \ldots, d_n$ számok legnagyobb közös osztója d, akkor

$$g(a_1, \ldots, a_n) \geq a_1 \left\lceil (a_1 - 1) d \frac{c_i}{d_i} \right\rceil + a_1 d - d - a_1,$$

ahol $\lceil w \rceil$ azt a legkisebb egészt jelöli, ami a w valós számnál nem kisebb.

6.10. *tétel* [8]. Legyenek adva az a_1, q_0, $q_1, \ldots, q_{n-2}$, $c_2, \ldots, c_n$ természetes számok, úgy, hogy a_1 és q_0 relatív prím. Jelölje $\alpha_2, \alpha_3, \ldots, \alpha_n$ az a_1 számjegyeit, a $q_1, \ldots, q_{n-2}$ által meghatározott általánosított számrendszerben, azaz

$$a_1 = \alpha_2 + \alpha_3 q_1 + \ldots + \alpha_n q_1 \ldots q_{n-2},$$

ahol

$$\begin{aligned} &\alpha_j \text{ egész} \quad j = 2, \ldots, n \\ &0 \leq \alpha_j < q_{j-1} \quad j = 2, \ldots, n-1 \end{aligned}$$

Tegyük fel, hogy k az a legkisebb index, melyre $\alpha_k > 0$.

Ha

$$a_j = c_j a_1 + q_0 q_1 \ldots q_{j-2}, \quad j = 2, \ldots, n$$

akkor

$$g(a_1, \ldots, a_n) = \max_{\substack{k \le t \le n \\ \alpha_t > 0}} [a_2(q_1 - 1) + \ldots + a_{t-1}(q_{t-2} - 1) + a_t(\alpha_t - 1) + a_{t+1}\alpha_{t+1} + \ldots + a_n \alpha_n] - a_1$$

Egy további alsó korlát található [12]-ben.

6.11. *tétel:* Tetszőleges relatív prím $a_1, \ldots, a_n$ természetes számokra

$$g(a_1, \ldots, a_n) \ge \frac{n-1}{n} \sqrt[n-1]{(n-1)! \prod_{j=1}^{n} a_j} - \sum_{j=1}^{n} a_j$$

Ugyanebben a dolgozatban található egy másik tétel, mely rámutat arra, hogy a $g(a_1, \ldots, a_n)$ szám nagyságának mennyire kell legalább meghaladni az $a_1, \ldots, a_n$ számok nagyságát.

6.12. *tétel:* Ha az $a_1, \ldots, a_n$ számok relatív prímek, akkor rögzített n esetén

$$1 \ge \liminf_{t \to \infty} \min_{a_1, \ldots, a_n \ge t} \frac{g(a_1, \ldots, a_n)}{(n-1)(\min_j a_j)^{1 + \frac{1}{n-1}}} \ge \frac{n-1}{e^n},$$

ahol e a természetes logaritmus alapja.

7. Polinomiális 0—1 probléma

A [20] dolgozat foglalkozik az alábbi nemlineáris problémával

$$\begin{aligned} &\max f(\mathbf{x}) \\ &g_i(\mathbf{x}) \le b_i \quad i = 1, \ldots, m \\ &\mathbf{x} \in \{0, 1\}^n, \end{aligned} \tag{7.1}$$

ahol az f és g_i $(i = 1, \ldots, m)$ függvények polinomok.

Elvben bármely bináris probléma leírható (7.1) alakban a következő tétel értelmében.

7.1 *tétel:* Ha $h(\mathbf{x})$ tetszőleges n-változós valós függvény, akkor létezik egy $P(\mathbf{x})$ valós együtthatós polinom, hogy

$$\forall \mathbf{x} \in \{0, 1\}^n \text{ esetén } h(\mathbf{x}) = P(\mathbf{x}).$$

Azonban a gyakorlatban a tétel nem alkalmazható, mert $P(\mathbf{x})$ zérustól különböző együtthatóinak száma általában igen nagy (legrosszabb esetben 2^n), így $P(\mathbf{x})$ kezelhetetlen. Mégis a gyakorlatban felmerülnek olyan problémák, amelyekre (7.1) megfelelő leírási forma.

A korábbi dolgozatok célja a (7.1) feladat linearizálása volt, melyet sikerült is elérniük számos új változó és feltétel bevezetése árán. Mivel azonban a megoldási idő a gyakorlatban elsősorban a változók számától függ, ezért az ilyen jellegű visszavezetések általában nem szerencsések.

A [20] dolgozat egyik fő eredménye az volt, hogy megmutatta, a 2. fejezetben tárgyalt leszámlálási módszerek egyszerű módosítással a (7.1) feladatra is alkalmazhatók. Egy általános leszámlálási struktúrát megvalósító programot sikerült is igen kis többletmunkával ennek a problémának a megoldására is alkalmassá tenni.

Végezetül tekintsük a feltétel nélküli feladatot,

$$\max \sum_{i=1}^{p} a_i \prod_{j \in Q_i} x_j$$
$$x \in \{0, 1\}^n, \tag{7.2}$$

ahol $Q_i \subset \{1, \ldots, n\}$, $i = 1, \ldots, p$. Magasabb hatványok nem szerepelnek, mert bináris változók esetén bármely k természetes száma $x_j^k = x_j$. Egyszerű szükséges feltétel adható meg arra nézvést, hogy egy pont (7.2) optimális megoldása legyen.

7.2 *tétel:* Válasszunk egy $\bar{x} \in \{0, 1\}^n$ pontot.
Legyen ekkor

$$A_j = \sum_{\substack{i: \\ j \in Q_i}} a_i \prod_{\substack{q \in Q_i \\ q \neq j}} \bar{x}_j. \quad j = 1, \ldots, n$$

Ha $\bar{x}$ optimális megoldása a (7.2) feladatnak, akkor

$$\bar{x}_j = 1 \text{ esetén } A_j \geq 0$$
$$\bar{x}_j = 0 \text{ esetén } A_j \leq 0.$$

8. Ütemezési problémák

A diszkrét programozásnál tágabban értelmezett kombinatorikus optimalizálás körébe tartoznak az ütemezési feladatok, ha az őket definiáló adatok determinisztikusak.

Az alább ismertetendő mindkét modellben egyetlen erőforrást kell szétosztani időben különböző feladatok között. A kettő között a különbség abban rejlik, hogy míg az első esetben az erőforrás oszthatatlan, vagyis egyszerre mindig egyetlen feladattal tud foglalkozni, addig a második esetben az erőforrás bizonyos kvantumokban feloszthető. Ezenfelül az utóbbi átfogalmazva bizonyos darabolási problémákat is megold.

8.1 *Egy egygépes ütemezési feladat*

A következőkben a [14] dolgozat eredményeit foglaljuk össze.

Egy gépen n munkadarabot kell megmunkálni. A megmunkálások között sorrendi megkötöttségek nincsenek. Ezzel szemben az i-edik munkadarab csak r_i időpontban áll rendelkezésre, határideje d_i és a szükséges megmunkálási

idő p_i. Az r_i időpontokat mint fizikai adottságokat fogjuk fel, tehát a megmunkálás semmiképpen sem kezdődhet r_i előtt, azonban d_i-nél befejeződhet később, de ezt nyilván szeretnénk elkerülni. Egy ütemezés azonosítható a munkadarabok egy sorrendjével, ha egy megmunkálást azonnal megkezdünk, ahogy arra lehetőség van. Ha c_i jelöli az i-edik munkadarab megmunkálásának befejezését, akkor a feladat feltételei a következők:

$$(i)\ c_i \geq r_i + p_i$$

$$(ii)\ \text{ha } i \neq j, \text{ akkor } [c_i - p_i, c_i) \cap [c_j - p_j, c_j) = \emptyset.$$

A cél a maximális késés minimalizálása, azaz

$$(iii)\ \min_{\pi:} \max_{i:} |c_i - d_i|_+,$$

ahol π a munkadarabok egy tetszőleges sorrendjét jelöli. Feltesszük továbbá, hogy a feladat adatai (r_i, p_i, d_i; $i = 1, \ldots, n$) nemnegatív egészek.

Az így definiált feladatra nagyobb méretekben csak lassú algoritmusokat sikerült megadni, ezért előtérbe került a heurisztikus eljárások vizsgálata. Számos ilyen eljárás, így az alábbiak is, az általuk javasolt sorrendet a megmunkálási idők figyelembevétele nélkül adják meg. Mint majd látni fogjuk, ez adott esetben különösebb hátrányt nem jelent.

8.1 *definíció:* Rossz párnak nevezünk egy (i, j) $(i \neq j)$ munkadarab párt, ha

$$r_i < r_j \text{ és } d_j < d_i.$$

8.2 *definíció:* Nagyon rossz párnak nevezünk egy (i, j) $(i \neq j)$ munkadarab párt, ha

$$r_i + p_i < r_j \text{ és } d_j < d_i.$$

8.3 *definíció:* Rossz hármasnak nevezünk egy (i, j, k) $(i \neq j, k; j \neq k)$ munkadarab hármast, ha (i, k) nagyon rossz pár és (j, k) rossz pár.

Schrage [18] nevéhez fűződik az alábbi elég természetes heurisztikus eljárás:

1. Elsőként végezzük el azt a munkát, amelyiket a legkorábban megkezdhetünk (vagyis azt, amelyikre r_i minimális). Ha több ilyen van, akkor ezek közül azt, amelyiknek a határideje a legrövidebb. Ha még mindig ilyen van, akkor ezek közül egyet választunk.

2. Ha a munka elvégzésekor egyetlen másikhoz sem foghatunk hozzá, akkor a következő munkát az 1. ponthoz hasonlóan választjuk ki. A két munka között a gép áll. Ellenkező esetben egy minimális határidejű munkát választunk.

3. A fenti eljárást mindaddig folytatjuk, amíg valamennyi munkát sorrendbe nem állítottuk.

Bárhogy adjunk is meg egy sorrendet, az ütemezést végezhetjük úgy, hogy az adott sorrendben azonnal felteszünk a gépre egy munkadarabot, mihelyt lehet. Ekkor a gép időnként áll, két állás között pedig folyamatosan működik, esetleg egymás után több munkadarabot is megmunkálva. Egy-egy ilyen működési periódusba eső munkadarabok összességét nevezzük egy blokknak. Az is nyilvánvaló, hogy a gép ilyen ütemezés mellett csak akkor áll, ha éppen

nem végezhet egyetlen munkát sem. Bevezetjük a következő jelölést. Legyen Π a munkák egy tetszőleges sorrendje.

Ekkor

$$T(\Pi) = \{i\colon (i, \Pi(n)) \text{ rossz pár; } i \text{ és } \Pi(n) \text{ egy blokkban van}\}, \quad (8.1)$$

továbbá $T(\Pi) \neq \emptyset$ esetén $j(\Pi)$ legyen az az index, melyre

$$c_{j(\pi)} = \max_{i \in T} c_i.$$

Nyilván $j(\Pi)$ egyértelműen meghatározott.

Az eljárás hibájára *Carlier* [9] adott korlátot.

8.1 *tétel:* Legyen Π a fenti algoritmussal szolgáltatott sorrend. Ekkor

(i) ha $T(\Pi) = \emptyset$, akkor Π optimális

(ii) ha $T(\Pi) \neq \emptyset$, akkor az eljárás hibája legfeljebb $p_{j(\pi)} - 1$.

8.2 *tétel:* Ha nincs rossz pár, akkor Schrage algoritmusa optimális megoldást ad.

8.3 *tétel:* A Scharge algoritmus lépésszáma $O(n \log n)$.

Az utóbbi abból következik, hogy az algoritmus lényegében a (d_i, r_i) párokat rendezi lexikografikus monoton növő sorrendbe. A 8.2. tétel pedig az előtte lévőnek azonnali következménye.

Most a Schrage algoritmus felhasználásával megadunk egy módosított eljárást [14].

1. A Schrage algoritmus alkalmazásával meghatározzuk a Π sorrendet.
2. Ha $T(\Pi) = \emptyset$, akkor megyünk a 3. lépésre. Különben legyen

$$r_{j(\pi)} = r_{\pi(n)}.$$

Megyünk az 1. lépésre

3. Az eddig generált sorrendek közül kiválasztjuk a legjobbat.

Erre az eljárásra igazak a következők.

8.4 *tétel:*

(i) A módosított eljárás hibája legfeljebb annyi, mint a Schrage eljárásé.

(ii) A módosított eljárás optimális megoldást adja, ha nincs rossz hármas.

(iii) A módosított eljárás lépésszáma legfeljebb $O(n^3 \log n)$.

Tehát sikerült azon feladatok körét, amelyekre a heurisztikus eljárás optimális megoldást ad, lényegesen kiterjeszteni, anélkül, hogy a számítási mennyiség reménytelenül megnőne. A gyakorlatban a (iii)-ban adott értéknél sokszor lényegesen kevesebb lépés is elég.

8.2. *Egy ütemezési feladat megoldása darabolási problémaként*

A [6] dolgozat foglalkozik az alábbi problémával. Adott egy erőforrás és bizonyos munkák, amelyek között azt szét kell osztani. Maga az erőforrás egy időpillanatban bizonyos egyenlő részekre, illetve annak többszöröseire felosztható. (Például ha az erőforrás egy brigád, akkor a brigád tagjai egyszerre több munkán is dolgozhatnak, de nyilván mindenki csak egyen.) Az egyes munkákra

meg van adva, hogy az erőforrásból hány egységnyi részt, időben milyen hosszan kötnek le. Mindkét adat rögzített. A feladat a munkák olyan ütemezésének elkészítése, ahol a teljes átfutási idő (az első munka megkezdésétől az utolsó munka befejezéséig számított idő) minimális. Ezt a célfüggvényt ütemezési feladatoknál előszeretettel alkalmazzák, mert megfelelően képvisel több más célt, pl. a határidők tartását is.

A probléma átfogalmazható darabolási problémává. Adott egy szalag (elegendően hosszú), melyből meghatározott méretű téglalapokat kell kivágni a szalag hosszával párhuzamosan. A téglalapok nem forgathatók. A feladat a téglalapok egy olyan elrendezését találni, mely a szalagból minimális hosszat foglal el. Ilyen problémához jutunk, ha az anyag különböző irányokban különbözően viselkedik (pl. az egyik oldallal párhuzamos szálakat tartalmaz).

A probléma matematikai leírása több lépcsőből áll. Először egy hálózatot definiálunk, majd pedig szükséges és elegendő feltételt adunk arra, hogy egy hálózatbeli folyam mikor jelöli ki a téglalapok egy elrendezését.

Adottak tehát a $T_1, \ldots, T_m$ téglalapok, ahol

$$T_i = k_i \times w_i, \quad i = 1, \ldots, m,$$

ahol k_i jelöli a téglalapnak a szalag hosszával párhuzamos méretét. A szalag szélessége w.

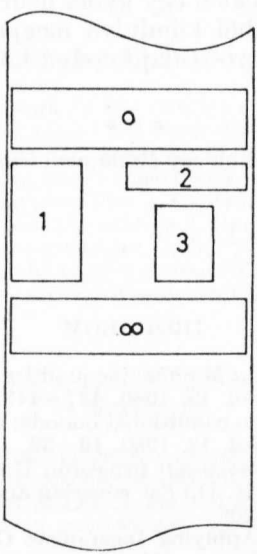
8.4 *definíció:* T-hálózatnak nevezzük a következőt:

(*i*) a gráf csúcsai

$$V = \{0, 1, \ldots, m, \infty\}$$

(*ii*) a gráf élei

$$E = \{(i, j): 1 \leq i, j \leq m, i \neq j\} \cup$$
$$\cup \{(0, i): 1 \leq i \leq m\} \cup \{(i, \infty): 1 \leq i \leq m\} \cup \{(0, \infty)\}$$



1. ábra

(*iii*) az élek kapacitásai

$$a_{ij} = \min\{w_i, w_j\} \quad 1 \leq i, j \leq m, \; i \neq j$$
$$a_{0i} = a_{i\infty} = w_i \quad 1 \leq i \leq m$$
$$a_{0\infty} = w.$$

Tekintsük most a téglalapokat egy tetszőleges elrendezését a szalagon és képzeljük úgy, hogy a szalag hossza függőleges irányú. Mivel az elrendezés mindenképpen véges helyet foglal el, ezért elhelyezhetünk két további, a szalag teljes szélességét elfoglaló téglalapot az elrendezés fölé (0) és alá (∞). Azt mondjuk, hogy egy téglalap egy másikon van, ha fölötte van és a szalag hosszára merőleges oldaluknak van olyan darabja, hogy ott a két téglalap között nincs további harmadik. Így az 1. ábrán 0 rajta van 1-en, 2-n és ∞-en; 2 pedig 3-on és ∞-en.

8.5 *definíció:* Tekintsünk egy elrendezést, melynek alapján a T hálózatban definiáljuk a következő ún. T-folyamot. Az (i, j) élen akkor folyik folyam, ha i a j-n van, és a folyam értéke megegyezik azzal a hosszal, amivel i a j téglalap van.

Nyilvánvaló, hogy a T-folyam értéke w és az egyes téglalapokra folyó (illetve onnan kimenő) folyam értéke pedig a téglalap szélessége.

8.6 *definíció:* T-gráf egy olyan része a T-hálózatnak, ahol a T-folyam értéke egy alkalmas elrendezés mellett pozitív.

8.5 *tétel:* Egy G összefüggő irányított gráfhoz, melynek van egy 0 kifokú és egy 0 befokú csúcsa, akkor és csak akkor található egy alkalmas téglalaprendszer és ennek egy olyan megfelelő elrendezése, hogy G az ehhez tartozó T-gráf, ha G nem tartalmaz irányított kört és síkba rajzolható.

Ennek a tételnek a segítségével egy gyors heurisztikus eljárás adható meg, mely egy konkrét elrendezésből kiindulva megpróbálja azt tovább javítani. A javíthatóság feltétele bizonyos tulajdonságú körök létezése a T-hálózatban.

* * *

A szerző köszönetét fejezi ki a cikk egy általa nem ismert lektorának alapos, a dolgozat jobbítását célzó véleményéért.

(Beérkezett: 1986. február 3-án.)

IRODALOM

1. Bachem, A. and Schrader, R.: Minimal inequalities and subadditive duality. *SIAM J. Control and Optimization*, Vol. 18. 1980. 437—443.
2. Balas, E.: Cutting planes from conditional bounds: a new approach for set covering. *Mathematical Programming*, Vol. 12. 1980. 19—36.
3. R. Bellman: *Dynamic Programming.* Princeton University Press, Princeton, 1957.
4. Bellmore, M. and Ratliff, H. D.: Set covering and involutory bases. *Management Science*, Vol. 18. 1971. 194—206.
5. Biró, M.: Efficient Method Applying Incomplete Ordering for Solving the Binary Knapsack Problem. in: K. Iracki, M. Malanowski and S. Walukiewicz (eds) *Optimization Techniques.* Springer-Verlag 1980. Part 2., 160—169.

6. Biró, M. & Boros, E.: A Network Flows and Non-Guillotine Cutting Patterns. *European Journal of Operational Research (EJOR), 16.* 1984. 215—221.
7. Boros E.: Egy új elv s-feltételek meghatározására. MTA SZTAKI *WP* MO/28.
8. Boros, E.: Subadditive Approach to a Linear Diophantine Problem of Frobenius. MTA SZTAKI *WP* MO/38.
9. Carlier, J.: The One-Machine Sequencing Problem. *European Journal of Operational Research*, Vol. 11. 1982. 42—47.
10. Everett, H.: Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation. *Operations Research*, Vol. 11. 1963. 399—417.
11. S. Halfin: Arbitrary Complex Corner Polyhedra Are Dense in R^n. *SIAM J. Appl. Math.*, Vol. 23 No. 2. September 1972. 157—163.
12. Hujter, M.: Lower bounds for the Frobenius problem. MTA SZTAKI *WP* MO/43.
13. Hujter, M.: On a problem of Frobenius: a survey. MTA SZTAKI *WP*. MO/44.
14. Hujter, M.: An effective algorithm for one-machine maximum lateness problem. Kézirat, publikálás alatt.
15. Kovács L. B.: Leszámlálási struktúrák és alkalmazásuk diszkrét programozási feladatok megoldására. *Matematikai Lapok*, XIX, 33—48.
16. Kovács, L. B.: Solution of Linear Integer Programming Problems by Dynamic Programming. *Math. Operationsforschung u. Statistik*, Vol. 5. 1974.
17. Kovács, L. B.: Combinatorial Methods of Discrete Programming. *Mathematical Methods of Operations Research 2*, ed.: A. Prékopa, Akadémiai Kiadó, Budapest 1980.
18. Schrage, L.: Obtaining optimal solutions to resource constrained network scheduling problems. Publikálatlan kézirat, 1971.
19. Sebő, A.: Brief Description of a Class of Cutting Planes for the Set CoveringProblem. MTA SZTAKI *WP* MO/42.
20. Vizvári B.: Leszámlálási algoritmusok a 0—1-es polinomáliás programozásban. *Alkalmazott Matematikai Lapok*, 1. 1975. 373—384.
21. Vizvári B.: A Lagrange szorzók használata diszkrét programozási algoritmusokban. *Alkalmazott Matematikai Lapok*, 2. 1976. 413—425.
22. Vizvári, B.: Lagrage Multipliers in Integer Programming *Problems of Control and Information Theory*, Vol. 7. 1978. 393—406.
23. Vizvári, B.: On the Connection of the Frobenius Problem and the Knapsack Problem. in: *Colloquia Mathematica Societatis János Bolyai*, 37. Finite and Infinite Sets, North-Holland, 799—819.
24. Vizvári, B.: An Application of Gomory Cuts in Number Theory. MTA SZTAKI *WP*. MO/40.

SOME RECENT HUNGARIAN RESULTS IN DISCRETE PROGRAMMING

The paper is devoted to the work in the field of optimization done by the discrete programming group of the Operations Research Department at the Computer and Automation Institute of the Hungarian Academy of Sciences. Results of the theoretical study of the sets of binary vectors, graph theory and other parts of combinatorics have been omitted. The titles of the chapters are: 1. General aspects of discrete programming problems 2. General methods of problem solving 3. Heuristic methods 4. Special problem classes 5. Duality, 6. Non-zero-one problems 7. The polynomial zero-one programming 8. Scheduling problems. Many of the results contained in the paper have not been published yet but in working papers, especially some in Chapters 4, 5, 6 and 8.

НОВЫЕ РЕЗУЛЬТАТЫ В ДИСКРЕТНОМ ПРОГРАММИРОВАНИИ В ВЕНГРИИ

Цель работы состоит в том, чтобы показать результаты, достигнутые группой дискретного программирования отдела исследования операций Института вычислительной техники и автоматизации ВАН (MTA SZTAKI). Многие из них до сих пор были опубликованы лишь в форме т. н. препринтов, поэтому в особенности разделы 4, 5, 6 и 8 содержат много нового.