Dr.Bajalinov Erik

Debreceni Egyetem Informatikai Kara
Bajalinov@Inf.UniDeb.Hu

Hiperbolikus programozás – Elmélet, módszerek, alkalmazások, szoftver

mobiDIÁK könyvtár

Tartalomjegyzék

I. Fejezet
Bevezetés a hiperbolikus programozásba

A *hiperbolikus programozási* (HP) feladat alatt olyan optimumszámítási feladatot értünk, amelyben a feltételek lineáris egyenlőség és egyenlőtlenség formájaban adottak, és ezen feltételek mellett keressük két lineáris függvény hányadosának a maximumát vagy minimumát.

A gyakorlatban a hiperbolikus programozási feladatok akkor vetődnek fel, amikor valami végrehajtandó tudatos tevékenység több módon is megvalósítható, és ezek módok (eljárások) közül ki kell választanunk egy olyat, amely mellett a szóban forgó tevékenységnek a fajlagos gazdasági mutatója a legmagasabb. Ezekből a fajlagos gazdasági mutatókból ebben a könyvben leggyakrabban a következőket használjuk : hatékonyság mint eredmény per kiadás (pl. profit per összes költség vagy összes előállított termék ára per felhasznált munkaidő), fajlagos önköltség mint összes költség per a termék egy egysége (pl. összes szállítási költség per a szállítandó termék mennyisége) stb.

Manapság a természetes anyagok és erőforrások korlátozottsága és drágasága miatt az ilyen fajta fajlagos mutatók használata a gazdasági életben már elkerülhetetlen.

1. Hiperbolikus programozási feladat

A hiperbolikus programozási feladatot Martos Béla vezette be az 1960-as években a [**133**], [**134**] cikkeiben, ahol megfogalmazta ezt a feladatot, megvizsgálta a feladat tulajdonságait és kidolgozott egy szimplex módszer alapú algoritmust a feladat megoldására.

A *hiperbolikus* kifejezés azért kapott helyet a feladat elnevezésében, mert egyváltozós esetben két lineáris függvény hányadosának grafikusan megfelel egy hiperbola. Amúgy a feladatnak ez a "grafikus" alapú elnevezése csak a magyar nyelvű szakirodalomban szokásos, az angol vagy orosz nyelvű szakirodalomban a feladatot "lineáris-tört programozási" feladatnak szokás

nevezni, azaz "*Linear-Fractional Programming*" vagy "Дробно-Линейное Программирование".

1.1. A feladat megfogalmazása

Tipikus esetben *általános* hiperbolikus programozási feladat alatt a következő feladatot értünk:

Adott

$$Q(x) = \frac{P(x)}{D(x)} = \frac{\sum_{j=1}^{n} p_j x_j + p_0}{\sum_{j=1}^{n} d_j x_j + d_0} \tag{1}$$

cél-függvény, amelyet kell maximalizálni vagy minimalizálni a következő feltételek mellett

$$\begin{cases} \sum_{j=1}^{n} a_{ij} x_j \le b_i, & i = 1, 2, \ldots, m_1, \\ \sum_{j=1}^{n} a_{ij} x_j \ge b_i, & i = m_1 + 1, m_1 + 2, \ldots, m_2, \\ \sum_{j=1}^{n} a_{ij} x_j = b_i, & i = m_2 + 1, m_2 + 2, \ldots, m, \end{cases} \tag{2}$$

$$x_j \ge 0, \quad j = 1, 2, \ldots, n_1, \tag{3}$$

ahol $m_1 \le m_2 \le m$, $n_1 \le n$.

Itt és továbbiakban feltételzünk, hogy $D(x) \neq 0$, $\forall x = (x_1, x_2, \ldots, x_n) \in S$, ahol S jelöli a(z) (2)-(3) feltételek által meghatározott *lehetséges megoldások halmazát* vagy a *lehetséges halmazt* (agolul – *"feasible set"*, oroszul – *"допустимое множество"*).

Mivel $D(x) \neq 0$ $\forall x \in S$, az általánosság megszorítása nélkül feltételezhetjük, hogy

$$D(x) > 0, \ \forall x \in S. \tag{4}$$

Abban az esetben, ha $D(x) < 0$, $\forall x \in S$, egyszerűen megszorozzuk a $P(x)$ és $D(x)$ függvényeket (-1)-gyel.

Itt és mindenütt a továbbiakban csak olyan hiperbolikus programozási feladatokkal foglalkozunk amelyek kielégítik a(z) (4) feltételt.

Továbbá feltételezzük, hogy a(z) (2) feltételrendszer minden feltétele lineárisan független, úgyhogy az $A = ||a_{ij}||_{m\times n}$ mátrix rangja pontosan m, azaz $rank(A) = m$.

Tehát a hiperbolikus programozási feladat esetén célunk az, hogy kell keresnünk olyan x vektort (vagy olyan $x_j,\ j = 1, 2, \ldots, n,$ változókat), amely

(1) maximálizálja (vagy minimálizálja) a $Q(x)$ *cél-függvény*t (az angol nyelvű szakirodalomban – *"objective function"*, oroszul pedig *"целевая функция"*), és
(2) kielégíti a(z) (2) *fő feltételeket* (*"main constraints"*, *"главные условия"*) és a(z) (3) *nem-negatívitási feltételeket* (*"sign restrictions"*, *"условия неотрицательности"*).

1.2. Fő definíciók

Most vezessük be a szükséges fogalmakat, definíciókat.

I.1. Definíció. *Ha egy adott $x = (x_1, x_2, \ldots, x_n)$ vektor kielégíti a(z) (2) és (3) feltételeket, azt fogjuk mondani, hogy az x vektor a(z) (1)-(3) hiperbolikus programozási feladatnak* ***lehetséges*** *(vagy* ***megengedett****)* ***megoldása*** *(angolul – "feasible solution", oroszul – "допустимое решение").*

I.2. Definíció. *Ha egy adott $x = (x_1, x_2, \ldots, x_n)$ vektor lehetséges megoldása a(z) (1)-(3) maximalizálási (minimalizálási) hiperbolikus programozási feladatnak, és az adott x pontban $Q(x)$ cél-függvény felveszi az S fölött a maximális (minimális) értékét, akkor az x vektort* ***optimális megoldásnak*** *nevezzük (angolul – "optimal solution", oroszul – "оптимальное решение").*

I.3. Definíció. *Azt mondjuk, hogy a(z) (1)-(3) maximalizálási (minimalizálási) hiperbolikus programozási feladat* ***megoldható*** *(angolul – "solvable", oroszul – "задача разрешима"), ha*

(1) *az S lehetséges halmaz nem üres, azaz $S \neq \emptyset$,*
(2) *az S lehetséges halmaz fölött a $Q(x)$ cél-függvény rendelkezik véges felső (alsó) korláttal.*

I.4. Definíció. *Ha az S lehetséges halmaz üres, azaz $S = \emptyset$, , akkor a feladatot (főleg az angol nyelvű szakirodalomban)* ***lehetetlennek*** *nevezik ("infeasible").*

I.5. Definíció. *Ha a maximalizálási (minimalizálási) hiperbolikus programozási feladatban cél-függvény felülről (alulról) nem korlátos, akkor a feladatot (főleg az angol nyelvű szakirodalomban)* ***nem korlátlannak*** *("unbounded") nevezik.*

I.6. Definíció. *Ha a hiperbolikus programozási feladat lehetetlen vagy korlátlan, akkor a feladatot* ***megoldhatatlannak*** *fogjuk nevezni (angolul – "unsolvable", oroszul – "неразрешимая задача").*

1.3. Kapcsolat a lineáris programózási feladattal

Könnyen észre lehet venni, hogy abban az esetben, amikor

$$d_j = 0, \quad j = 1, 2, \dots n, \quad \text{és} \quad d_0 = 1, \tag{5}$$

akkor a(z) (1)-(3) hiperbolikus programozási feladat lineáris programozási feladattá válik.

Ez az oka és magyarázata annak, hogy a(z) (1)-(3) hiperbolikus programozási feladatot gyakran a lineáris programozási feladat általánosításának tekintik. Valóban, ha teljesülnek a(z) (5) feltételek, akkor az eredeti (1)-(3) hiperbolikus programozási feladatból a következőt kapjuk:

$$P(x) = \sum_{j=1}^{n} p_j x_j + p_0 \to \max(\text{ vagy } \min) \tag{6}$$

a következő feltételek mellett

$$\begin{cases} \sum_{j=1}^{n} a_{ij} x_j \le b_i, & i = 1, 2, \dots, m_1, \\ \sum_{j=1}^{n} a_{ij} x_j \ge b_i, & i = m_1 + 1, m_1 + 2, \dots, m_2, \\ \sum_{j=1}^{n} a_{ij} x_j = b_i, & i = m_2 + 1, m_2 + 2, \dots, m, \end{cases} \tag{7}$$

$$x_j \ge 0, \quad j = 1, 2, \dots, n_1, \tag{8}$$

Ezen kívül van néhány speciális eset, amikor az eredeti hiperbolikus programozási feladat redukálódik (mondhatjuk úgy is, hogy "helyettesíthető") a hozzáillő lineáris programozási feladattá:

(1) Ha $d_j = 0, \ j = 1, 2, \dots n, \ \text{és} \ d_0 \neq 0,$ akkor $Q(x)$ cél-függvény lineárissá válik:

$$Q(x) = \sum_{j=1}^{n} \frac{p_j}{d_0} x_j + \frac{p_0}{d_0} = \frac{P(x)}{d_0}.$$

Ebben az esteben az eredeti cél-függvény maximalizálása (minimalizálása) helyettesíthető a $P(x)/d_0$ lineáris függvény maximalizálásával (minimalizálásával) az eredeti S lehetséges halmazon.

(2) Ha $p_j = 0,\ j = 1, 2, \dots n,$ és $p_0 \neq 0$, akkor

$$Q(x) = \frac{P(x)}{D(x)} = \frac{p_0}{\sum_{j=1}^{n} d_j x_j + d_0}$$

cél-függvény helyettesíthető a $D(x)$ függggvénnyel. Ebben az esetben az eredeti cél-függvény maximalizálása (minimalizálása) helyettesíthető a $D(x)$ függvény maximalizálásával vagy minimalizálásával (a p_0 előjelétől függően) az eredeti S lehetséges halmazon.

Világos, hogy ha $p_0 = 0$, akkor az egész feladat értelmetlenné válik, mivel ilyenkor a $Q(x)$ függvény a nulla értékű konstanssá válik.

(3) Ha $p = (p_1, p_2, \dots, p_n)$ és $d = (d_1, d_2, \dots, d_n)$ olyanok, hogy létezik olyan $\mu \neq 0$, hogy $p = \mu d$, akkor a

$$Q(x) = \frac{P(x)}{D(x)} = \frac{\sum_{j=1}^{n} \mu d_j x_j + p_0}{\sum_{j=1}^{n} d_j x_j + d_0} = \dots = \mu + \frac{p_0 - \mu d_0}{\sum_{j=1}^{n} d_j x_j + d_0} \tag{9}$$

cél-függvényt lehet helyettesíteni a $D(x)$ függvénnyel. Mégpedig úgy, hogy az eredeti $Q(x)$ cél-függvény maximalizálása (minimalizálása) vezet a

- $D(x)$ függvény minimalizálásához (maximalizálásához) ha $p_0 - \mu d_0 > 0$,
- $D(x)$ függvény maximalizálásához (minimalizálásához) ha $p_0 - \mu d_0 < 0$.

Itt meg kell jegyeznünk, hogy abban az esetben, ha $p_0 - \mu d_0 = 0$, a(z) (9) képletnek megfelelően $Q(x) = \mu,\ \forall x \in S$, és az egész feladat értelmetlenné válik.

A továbbiakban kizárjuk a vizsgálatból a következő eseteket:

(1) $P(x) = const,\ \forall x \in S$;
(2) $D(x) = const,\ \forall x \in S$;
(3) $Q(x) = const,\ \forall x \in S$,;

mivel ezekben az esetekben az eredeti hiperbolikus programozási feladat vagy helyettesíthető megfelelő lineáris programozási feladattal (az első két eset), vagy teljesen elveszíti az értelmét. (3. eset).

1.4. A Hiperbolikus programozási feladat fő alakjai

A előző részben láttuk, hogy a hiperbolikus programozási feladathoz tartozó feltételrendszer tartalmaz(hat) lineáris egyenlőségeket és egyenlőtlenségeket. Ezenkívül láttuk azt is, hogy a nem-negativitási feltételek nem mindig az összes ismeretlen változóra vonatkoznak. Tehát, a hiperbolikus programozási feladat tartalmazhat az alulról korlátlan változókat is, azaz olyan változókat, amelyek felvehetnek negatív értéket is (angolul – "*unrestricted in sign variable*" vagy "*urs variable*", oroszul – *"переменная неограниченная снизу"*).

Mielőtt továbbmegyünk, tekintsünk meg néhány speciális alakú hpi feladatot, és tanuljuk meg, hogyan lehet ezeket a különböző alakú hiperbolikus programozási feladatokat egymásba átalakítani. Erre azért van szükség, mert leggyakrabban a hpi feladat megoldására alkalmazható módszerek speciális alakú feladatot igényelnek.

I.7. Definíció. *A hiperbolikus programozási feladat* ***kanonikus alakú*** *(angoul – "canonical form", oroszul pedig – "каноническая задача"), ha a fő feltételrendszere csak egyenlőségekből áll, és minden ismertelen változó nemnegatív értékű:*

$$Q(x) = \frac{P(x)}{D(x)} = \frac{\sum_{j=1}^{n} p_j x_j + p_0}{\sum_{j=1}^{n} d_j x_j + d_0} \longrightarrow \max(\min), \tag{10}$$

a következő feltételek mellett

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1, 2, \ldots, m, \tag{11}$$

$$x_j \geq 0, \quad j = 1, 2, \ldots, n, \tag{12}$$

ahol $D(x) > 0$, $\forall x \in S$.

I.8. Definíció. *Az adott hiperbolikus programozási feladat* ***szabványos*** *vagy* ***standard alakú*** *(angolul – "standard form", oroszul – "стандартная задача"), ha a fő feltételrendszerében csak '$\leq$' ('kisebb vagy egyenlő') relációjú egyenlőtlenségek vannak, és minden ismertelen változó nem-negatív értékű:*

$$Q(x) = \frac{P(x)}{D(x)} = \frac{\sum_{j=1}^{n} p_j x_j + p_0}{\sum_{j=1}^{n} d_j x_j + d_0} \rightarrow \max(\min),$$

a következő feltételek mellett

$$\sum_{j=1}^{n} a_{ij}x_j \le b_i, \quad i = 1, 2, \ldots, m,$$

$$x_j \ge 0, \quad j = 1, 2, \ldots, n,$$

ahol $D(x) > 0$, $\forall x \in S$.

Nyilvánvaló, hogy a standard és a kanonikus HP feladat az általános (1)-(3) feladatnak speciális esetei.

Valóban, ha az általános (1)-(3) feladatban $m_1 = m_2 = 0$ és $n_1 = n$, akkor kapjuk a kanonikus feladatot. Abban az esetben, ha $m_1 = m$ and $n_1 = n$, az általános feladat standard feladattá válik.

Ezeket a különböző alakú feladatokat a megfelelő eljárások felhasználásával könnyen át lehet alakítani egymásba. Tekintsük ezeket az eljárásokat:

(1) '$\ge$' ('nagyobb mint') $\rightarrow$ '$\le$' ('kisebb mint').
A '$\ge$' relációjú feltétel mindkét oldalát meg kell szoroznunk (-1)-gyel.

(2) '$\le$' ('kisebb mint') $\rightarrow$ '=' ('egyenlő').
Be kell vezetni nem-negatív értékű "*mesterséges*" (angolul – "*slack variable*, szószerint "*holtjáték változó*", oroszul – *"исскуственная переменная"*) változót a '$\le$' relációjú feltétel bal oldalán úgy, hogy a mesterséges változó fogja játszani a bal és jobb oldal közötti különbség szerepét:

$$\sum_{j=1}^{n} a_{ij}x_j \le b_i \quad \rightarrow \quad \begin{cases} \sum_{j=1}^{n} a_{ij}x_j + s_i = b_i, \\ s_i \ge 0. \end{cases}$$

(3) Az alulról korlátlan x_j változó $\rightarrow$ nem-negatív x_j változó ($x_j \ge 0$).
Az x_j változó helyett be kell vezetni két darab nem-negatív x'_j és x''_j változót, majd mindenütt, ahol előfordul az x_j változó, azt helyettesítenünk kell az $(x'_j - x''_j)$ kifejezéssel, azaz:

$$x_j \;=\; x'_j \;-\; x''_j.$$

Majd a nem-negativítási feltételekhez hozzá kell adnunk a következő új feltételeket:

$$x'_j \ge 0 \quad \text{és} \quad x''_j \ge 0\,.$$

Mivel a HP feladat mindhárom alakja (általános, standard és kanonikus) könnyen átalakítható egymásba, ezért az egyszerűség kedvéért néha az általános feladat helyett a vele ekvivalens standard vagy kanonikus alakú feladatot fogjuk tanulmányozni.

Vezessük be a következő jelöléseket:

$$A_j = (a_{1j}, a_{2j}, \ldots, a_{mj})^T, \quad j = 1, 2, \ldots, n,$$
$$b = (b_1, b_2, \ldots, b_m)^T$$
$$A = (A_1, A_2, \ldots, A_n) = ||a_{ij}||_{m\times n}$$
$$x = (x_1, x_2, \ldots, x_n)^T$$
$$p = (p_1, p_2, \ldots, p_n)$$
$$d = (d_1, d_2, \ldots, d_n)$$

A jelölések felhasználásával a HP feladatot megfogalmazhatjuk mátrixos formában:

Kanonikus feladat:

$$Q(x) = \frac{px + p_0}{dx + d_0} \to \max,$$

a következő feltételek mellett

$$\sum_{j=1}^{n} A_j x_j = b \ (\text{ vagy } Ax = b \ )$$

$$x \geq 0,$$

ahol $D(x) = dx + d_0 > 0, \ \forall x \in S$.

Standard feladat:

$$Q(x) = \frac{px + p_0}{dx + d_0} \to \max,$$

a következő feltételek mellett

$$\sum_{j=1}^{n} A_j x_j \leq b \ (\text{ vagy } Ax \leq b \ )$$

$$x \geq 0,$$

ahol $D(x) = dx + d_0 > 0, \ \forall x \in S$.

Meg kell jegyeznünk, hogy a matematikai programozás elméletének megfelelően

$$\min_{x\in S} F(x) \equiv \max_{x\in S}(-F(x)). \tag{13}$$

Ez azt jelenti, hogy a minimalizálási hiperbolikus programozási feladatot helyettesíthetjük maximalizálási feladattal, csak ehhez az eredeti cél-függvényt meg kell szoroznunk (-1)-gyel.

Ezért a továbbiakban csak maximalizálási feladattal foglalkozunk.

2. Grafikus módszer

A következőkben a kétváltozós hiperbolikus programozási feladatok megoldásához alkalmazható grafikus módszert ismertetjük. A módszer azon alapul, hogy az egyenlőtlenség típusú feltételek félsíkokat határoznak meg a kétdimenziós térben, és ezen félsíkok metszeteként előállítható a lehetséges halmaz, melynek ismeretében meghatározható a feladat optimális megoldása vagy megmutatható, hogy a feladat nem megoldható.

2.1. Elmélet

Tekintsük a következő kétváltozós hiperbolikus programozási feladatot:

$$Q(x) \;=\; \frac{P(x)}{D(x)} \;=\; \frac{p_1x_1 + p_2x_2 + p_0}{d_1x_1 + d_2x_2 + d_0} \longrightarrow \max$$

a következő feltételek mellett

$$a_{i1}\, x_1 + a_{i2}\, x_2 \;\leq\; b_i, \quad i = 1, \ldots, m,$$

$$x_1 \geq 0, \;\; x_2 \geq 0\ .$$

Már tudjuk, hogy a lehetséges megoldások S halmaza egy konvex sokszög vagy egy olyan konvex (korlátos vagy korlátlan) halmaz, amelynek véges sok csúcspontja van. Ha $S = \emptyset$, akkor a definició szerint a feladat nem megoldható, azaz a feladatnak nem létezik optimális megoldása. Így tegyük fel, hogy $S \neq \emptyset$.

Egy további fontos kikötés annak feltételezése, hogy a

$$D(x) \;=\; d_1x_1 + d_2x_2 + d_0$$

függvény az S lehetséges halmazon nem veszi fel a 0 értéket, ezt a továbbiakban feltételezzük. Továbbá, mivel $D(x) \neq 0,\ \forall x \in S$, ezért az általánosság megszorítása nélkül feltételezhetjük, hogy a leheteséges megoldások halmazán a $D(x)$ nevező szigorúan pozitív, azaz $D(x) > 0,\ \ \forall x \in S$ (ellenkező esetben szorozzuk meg a $P(x)$ és $D(x)$ függvényeket (-1)-gyel.

2.1.1. *Speciális esetek.* Ebben a szekcióban soroljuk fel azokat a speciális esteket, amikor az eredeti megoldandó hiperbolikus programozási feladat "degenerált", és nem igényli a hiperbolikus programozási matematikai apparátus alkalmazását. A továbbikban ilyen feladatokkal nem foglalkozunk.

Mindenekelőtt vegyük észre, hogy amennyiben a

$$\begin{pmatrix} p_1 & p_2 \\ d_1 & d_2 \end{pmatrix}$$

mátrix determinánsa egyenlő 0-val, akkor a feladatnak triviális megoldása van, vagy a feladat visszavezethető egy alkalmas lineáris programozási feladat megoldására. A tekintett determináns értéke akkor és csak akkor 0, ha legalább az egyik sorvektor a $\mathbf{0}$ vektor, vagy a két sorvektor egymás 0-tól különböző konstansszorosa. Vizsgáljuk meg most ezeket az eseteket.

(1) Tegyük fel, hogy $(p_1, p_2) = \mathbf{0}$. Ekkor

$$Q(x) = \frac{p_0}{d_1\, x_1 + d_2\, x_2 \;+\; d_0}\ .$$

Ha $p_0 = 0$, akkor $Q(x) = 0$, minden $x \in S$-re, és így a feladatnak nincsen értelme.
Ha $p_0 > 0$, akkor a $Q(x)$ célfüggvény maximuma az S lehetséges halmazon megegyezik a $D(x) = (d_1x_1 + d_2x_2 + d_0)$ lineáris függvény S-en felvett minimumával.
Ha $p_0 < 0$, akkor a $Q(x)$ célfüggvény maximuma az S lehetséges halmazon megegyezik a $D(x) = (d_1x_1 + d_2x_2 + d_0)$ lineáris függvény S-en felvett maximumával.
Vegyük észre, hogy az utolsó két esetben lineáris programozási feladattal van dolgunk.

(2) Legyen $(d_1, d_2) = \mathbf{0}$. Ilyenkor, mivel $S \neq \emptyset$ és az S lehetséges halmazon

$$D(x) = d_1x_1 + d_2x_2 + d_0 > 0,$$

ezért $d_0 > 0$. Így,

$$Q(x) \;=\; \frac{p_1x_1 + p_2x_2 + p_0}{d_0} \;=\; \frac{p_1}{d_0}x_1 + \frac{p_2}{d_0}x_2 + \frac{p_0}{d_0}\ ,$$

azaz ebben az esetben $Q(x)$ célfüggvény lineáris, és ezért a feladat megoldásához lineáris programozási módszerekre van szükségünk.

(3) Tekintsünk az utolsó esetet: $(p_1, p_2) = \lambda\,(d_1, d_2)$. Ekkor bármely $x \in S$-re

$$\begin{aligned} Q(x) \;&=\; \frac{p_1x_1 + p_2x_2 + p_0}{d_1x_1 + d_2x_2 + d_0} \;=\; \frac{(\lambda\, d_1)\, x_1 + (\lambda\, d_2)\, x_2 + p_0}{d_1x_1 + d_2x_2 + d_0} = \\ &=\; \frac{(\lambda\, d_1)x_1 + (\lambda\, d_2)x_2 + p_0 + \lambda d_0 - \lambda d_0}{d_1x_1 + d_2x_2 + d_0} = \\ &=\; \frac{\lambda D(x) + p_0 - \lambda d_0}{d_1x_1 + d_2x_2 + d_0} \;=\; \lambda + \frac{p_0 - \lambda d_0}{d_1x_1 + d_2x_2 + d_0}\ . \end{aligned}$$

Másrészt a kapott célfüggvény formája azt mutatja, hogy ebben az esetben az eredeti maximalizálási hiperbolikus programozási feladat helyett a $D(x)$ lineáris függvény minimumát (ha $p_0 - \lambda d_0 > 0$)

vagy maximumát (ha $p_0 - \lambda d_0 < 0$) kell keresnünk az S halamzon. Nyilvánvaló, ha $p_0 - \lambda d_0 = 0$, akkor a feladatnak nincs értelme, mivel $Q(x) = \lambda \, , \forall x \in S$.

Következésképpen ha a $\begin{pmatrix} p_1 & p_2 \\ d_1 & d_2 \end{pmatrix}$ mátrix determinánsa 0, akkor a feladat megoldható egy alkalmas lineáris programozási feladat grafikus megodásával. Ezek után tegyük fel, hogy a kérdéses mátrix determinánsa 0-tól különböző.

2.1.2. *Általános eset - nívóvonalak és fókuszpont.* Vegyünk egy tetszőleges K konstanst. Azon x-ek halmazát, amely pontokban $Q(x) = K$ teljesül, a K konstanshoz tartozó *nívóvonalnak* nevezzük. Szemléletesen ez a felület azonos magasságú pontjait írja le. Térképészetben a különböző értékű nívóvonalak megadásával jelenítik meg a domborzati viszonyokat. Most megmutatjuk, hogy esetünkben a nívóvonalak egyenesek lesznek. Valóban, ha $Q(x) = K$, akkor

$$Q(x) \; = \; \frac{p_1x_1 + p_2x_2 + p_0}{d_1x_1 + d_2x_2 + d_0} = K \, ,$$

azaz

$$p_1x_1 + p_2x_2 + p_0 = K(d_1x_1 + d_2x_2 + d_0)$$

és x kielégíti a

$$(p_1 - Kd_1)x_1 + (p_2 - Kd_2)x_2 = Kd_0 - p_0$$

egyenletet. Mivel a kapott egyenlet lineáris, ezért azok az x pontok, amelyek kielégítik az adott lineáris egyenletet, egy egyenest alkotnak. Általános esetben minden külön K értékhez más-más egyenes nívóvonal tartozik.

Egy másik fontos észrevétel, hogy a nívóvonalakból álló egyenessereg egyenesei egy közös pontban, az úgynevezett *fókuszpontban* metszik egymást, ami az

$$\begin{cases} P(x) &= 0 \\ D(x) &= 0 \end{cases} \quad \text{azaz} \quad \begin{cases} p_1x_1 + p_2x_2 + p_0 &= 0 \\ d_1x_1 + d_2x_2 + d_0 &= 0 \end{cases}$$

egyenletrendszer által meghatározott egyenesek $F = (x_1, x_2)$ metszéspontja. Ez a metszéspont létezik, mivel feltettük, hogy a $\begin{pmatrix} p_1 & p_2 \\ d_1 & d_2 \end{pmatrix}$ mátrix determinánsa 0-tól különböző.

Továbbá a $d_1x_1 + d_2x_2 + d_0 \neq 0, \; \forall x \in S$ feltétel miatt $F \notin S$. Ahhoz, hogy a nívóvonalak egy pontban metsszék egymást, elegendő megmutatni, hogy minden nívóvonal átmegy az F ponton. Ez viszont triviális, mivel tetszőleges K-ra az F pont kielégíti az

$$p_1x_1 + p_2x_2 + p_0 \; = \; K(d_1x_1 + d_2x_2 + d_0)$$

egyenletet, hiszen az egyenletben szereplő lineáris kifejezés mindkét oldalon 0 értéket vesz fel az F pontban. Következésképpen a nívóvonalak egy pontban, a fókuszpontban metszik egymást.

Ezek után vizsgáljuk meg, hogy miként változik a nívóvonalakat meghatározó K érték az x_1 változó függvényében.

Válasszunk tetszőleges K értéket, ábrazoljuk a megfelelő $Q(x) = K$ egyenest (lásd 1. ábrát). És most írjuk át a $Q(x) = K$ egyenletet a következő formába:

$$x_2 = -\frac{p_1 - Kd_1}{p_2 - Kd_2}x_1 - \frac{p_0 - Kd_0}{p_2 - Kd_2}.$$

Látható, hogy a $Q(x) = K$ egyenletű nívóvonal α irányszögének

$$k = -\frac{p_1 - Kd_1}{p_2 - Kd_2}$$

tangense függ a K értéktől, és monoton, mert

$$\frac{dk}{dK} = \frac{d_1p_2 - d_2p_1}{(p_2 - Kd_2)^2}. \tag{14}$$

Továbbá a $\frac{dk}{dK}$ érték előjele nem függ a K értéktől, szóval

$$\mathsf{sign}\,\{\frac{dk}{dK}\} = \mathsf{sign}\,\{d_1p_2 - d_2p_1\} = const.$$

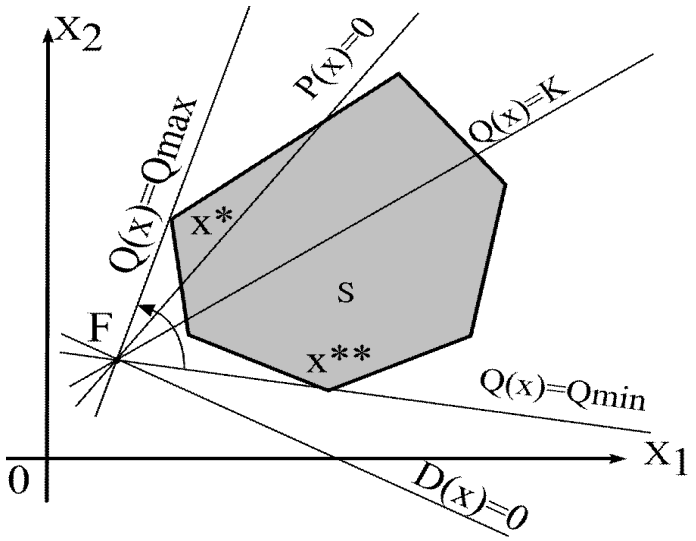
Ez pedig azt jelenti, hogy ha a nívóvonalat elforgatjuk az F fókuszpont körül pozitív irányba, akkor a $Q(x)$ értéke növekszik vagy csökken, attól függően, hogy milyen előjelű a $(d_1p_2 - d_2p_1)$ kifejezés. A nívóvonalat addig kell forgatnunk, amíg még lesz közös pontja a lehetséges megoldások halmazával. Világos, hogy, 1.ábra jelképezi azt az esetet, amikor a nívóvonal pozitív irányban történő elforgatása növeli a célfüggvény értékét.

Vegyük észre, hogy az F fókuszpont körül történő nívóvonal-forgatás során az adott esetben kapjuk a két szélső pontot – az egyik x^*, itt a cél-függvény a maximalis értéket veszi fel, a másik pedig x^{**}, amelyben a $Q(x)$ a minimális értéket feszi fel.

Most pedig soroljuk fel azokat a lehetséges eseteket, amelyek előfordulhatnak a grafikus módszer alkalmazása során:

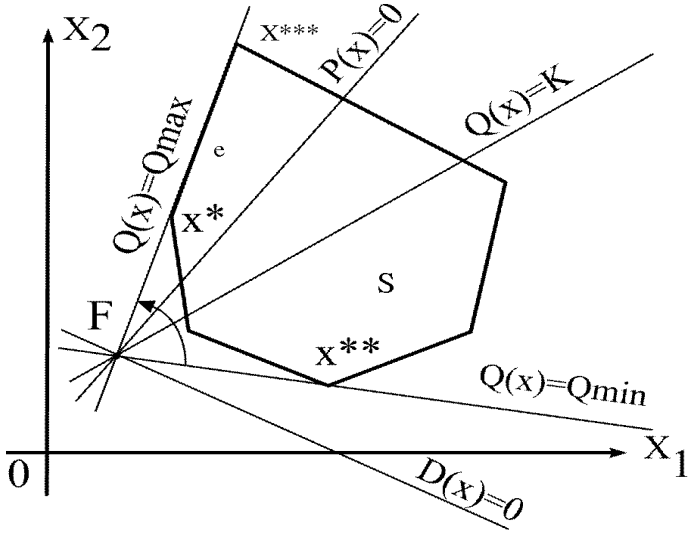
1. Létezik egyetlen egy olyan $x^* \in S$ pont (a lehetséges halmaz csúcspontja), hogy az x^* ponthoz tartozó nívóvonal egyetlen egy pontban metszi a lehetséges megoldások halmazát, és ez a pont éppen x^*. Ekkor ez a pont a feladat egyetlen optimális megoldása (1.ábra).

2. Létezik olyan $x^* \in S$ pont (a lehetséges halmaz csúcspontja), hogy az x^* ponthoz tartozó nívóvonal egy szakaszt metsz ki a lehetséges megoldások



1. ábra. Kétváltozós HP feladat – Egyetlen optimális megoldás.

halmazából. Ekkor a metszet minden pontja optimális megoldása a feladatnak, és csak ezek az optimális megoldások (2. ábra). Ebben az esetben a

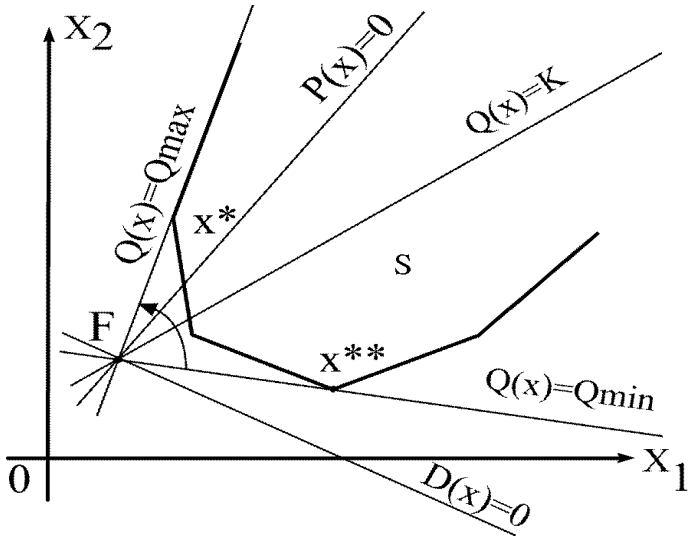


2. ábra. Kétváltozós HP feladat – Végtelen sok véges optimális megoldás.

feladat végtelen sok optimális megoldással rendelkezik (minden x pontja az e élnek). Ilyenkor az összes optimális megoldás előállítható a két csúcspont lineáris kombinációjaként:

$$x = \lambda x^* + (1-\lambda)x^{***}, \quad 0 \le \lambda \le 1.$$

3. Létezik olyan $x^* \in S$ pont (a lehetséges halmaz csúcspontja), hogy az x^* ponthoz tartozó nívóvonal egy félegyenest metsz ki a lehetséges megoldások halmazából. Ekkor a metszet minden pontja optimális megoldása a feladatnak, és csak ezek az optimális megoldások (3. ábra). Ebben az eset-



3. ábra. Kétváltozós HP feladat – Véges és végtelen optimális megoldások.

ben feladat végtelen sok optimális megoldással rendelkezik, és a megoldások között vannak véges pontok és vannak végtelen pontok (az összes x pontja a végtelen élnek).

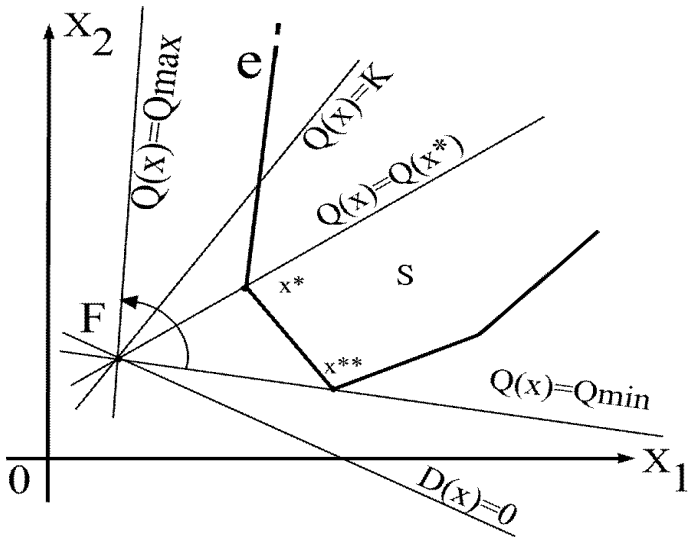
4. A nívóvonal forgatásakor mindig lesz a megfelelő nívóvonalnak metszete a lehetséges megoldások halmazával. Ekkor az x_1 és x_2 változók bármilyen nagy értéket vehetnek fel. Ez azt eredményezi, hogy a $Q(x)$ célfüggvénynek lehet véges vagy végtelen limesze az S halmazon (4. ábra).

A továbbiakban a lehetséges négy esetet és az eljárás alkalmazását a következő néhány numerikus példákon mutatjuk be.

2.2. Numerikus példák

Az **1.** eset illusztrálására tekintsük az alábbi hiperbolikus programozási feladatot:

$$Q(x) \;=\; \frac{2x_1 + x_2 - 5}{-2x_1 + x_2 + 7} \longrightarrow \max$$



4. ábra. Kétváltozós HP feladat – Aszimptótikus eset.

$$\begin{aligned} x_1 \qquad\quad &\leq 4 \\ -x_1 + x_2 &\leq 4 \\ x_1 + x_2 &\leq 8 \\ x_1 - x_2 &\leq 2 \end{aligned}$$

$$x_1 \geq 0, \quad x_2 \geq 0.$$

Meghatározva a lehetséges megoldások halmazát, azt kapjuk, hogy S egy olyan hatszög, amelynek csúcspontjai a $(0,0)$, $(2,0)$, $(4,2)$, $(4,4)$, $(2,6)$ és $(0,4)$ pontok. Felvéve a $-2x_1 + x_2 + 7 = 0$ egyenlet által meghatározott egyenest, ez az egyenes nem metszi a lehetséges megoldások halmazát, azaz a cél-függvény nevezője az S halmazon nem veszi fel a 0 értéket. Véve ennek az egyenesnek a metszetét a $2x_1 + x_2 - 5 = 0$ egyenlet által meghatározott egyenessel, megkapjuk az $F = (3,-1)$ fókuszpontot. Ezek után a(z) (14) képletnek megfelelően számóljuk ki a $(d_1p_2 - d_2p_1)$ kifejezés értékét:

$$d_1p_2 - d_2p_1 = -2 \times 1 - 1 \times 2 = -2 - 2 = -4 < 0\ .$$

Az utóbbi azt jelenti, hogy $\dfrac{dk}{dK} < 0$ és következésképpen a $Q(x)$ cél-függvény értékének növeléséhez a nyívóvonalat az óramutató járása irányában kell forgatnunk. Így nyílvánvalóvá válik, hogy a feladat optimlais megoldása a $(4,2)$ pontban van, úgy hogy $Q(x)$ cél-függvény optimális (azaz maximális) értéke

$$Q(4,2) \ = \ \frac{2 \times 4 + 2 - 5}{-2 \times 4 + 2 + 7} \ = \ \frac{5}{1} \ = \ 5\ .$$

A **2.** eset bemutatására vizsgáljuk a következő hiperbolikus programozási feladatot:

$$\begin{aligned} Q(x) \;=\; & \frac{x_1 + 2x_2 - 2}{2x_1 + 2x_2 - 2} \longrightarrow \max \\ -2x_1 \;+\; & 3x_2 \;\leq\; 3 \\ 3x_1 \;+\; & x_2 \;\geq\; 12 \\ 2x_1 \;-\; & 5x_2 \;\leq\; 8 \\ x_1 \;+\; & x_2 \;\leq\; 11 \\ & x_1 \geq 0, \;\; x_2 \geq 0 \end{aligned}$$

Meghatározva az S lehetséges halmazt, azt kapjuk, hogy S a $(4,0)$, $(9,2)$, $(6,5)$ és $(3,3)$ pontok által meghatározott négyszög, továbbá $F = (0,1)$. Felvéve a $2x_1 + 2x_2 - 2 = 0$ egyenletű egyenest, ez nem metszi a lehetséges megoldások halmazát, így a cél-függvény nevezője 0-tól különböző az S halmazon. Most számoljuk ki a $(d_1p_2 - d_2p_1)$ kifejezés értékét:

$$d_1p_2 - d_2p_1 = 2 \times 2 - 2 \times 1 = 4 - 2 = 2 > 0 \,.$$

Ebből következik, hogy $Q(x)$ a k lejtőszögnek egy monoton növekvő függvénye. Így addig kell forgatnunk a nyívóvonalat az óramutató járásával ellenkező irányban, amig nyívóvonalnak van még metszete az S halmazzal. Nyilvánvaló, hogy a keresett nyívóvonal az lesz, amelyik az F és a $(3,3)$ pontokat köti össze. Ebben az esetben a tekintett nívóvonal és S metszete a $(3,3)$ és $(6,5)$ pontokat összekötő szakasz. Ennek a szakasznak minden pontja optimális megoldás, és az optimum értéke pedig

$$Q(3,3) \;=\; \frac{3 + 2 \times 3 - 2}{2 \times 3 + 2 \times 3 - 2} \;=\; \frac{7}{10}$$

és

$$Q(6,5) \;=\; \frac{6 + 2 \times 5 - 2}{2 \times 6 + 2 \times 5 - 2} \;=\; \frac{14}{20} \;=\; \frac{7}{10} \,.$$

A **3.** eset szemléltetésére tekintsük az alábbi hiperbolikus programozási feladatot:

$$Q(x) \;=\; \frac{-x_1 - x_2 - 2}{2x_1 + 3x_2 + 3} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{aligned} -x_1 + x_2 &\leq 4 \\ x_1 \geq 0, \;\; & x_2 \geq 0 \end{aligned}$$

Most a lehetséges megoldások halmaza az a nemnegatív pontokat tartalmazó korlátlan konvex halmaz, amelyet az $x_1 = 0$, $x_2 = 4 + x_1$ egyenletek által meghatározott egyenesek és a $(0,0)$ és $(0,4)$ pontokat összekötő szakasz határolnak. Ha ábrázoljuk a $2x_1 + 3x_2 + 3 = 0$ egyenletű egyenest,

kiderül hogy az nem metszi a lehetséges megoldások halmazát, így a $Q(x)$ függvény nevezője 0-tól különbözik az S halamzon. Metszve ezt az egyenest a $-x_1 - x_2 - 2 = 0$ egyenletű egyenessel, megkapjuk az $F = (-3, 1)$ fókuszpontot.

Mivel

$$d_1 p_2 - d_2 p_1 = 2 \times -1 - 3 \times -1 = -2 + 3 = 1 > 0 \,,$$

ebből adódik, hogy $Q(x)$ cél-függvény az α lejtőszögnek egy monoton növekvő függvénye és addig kell forgatnunk a nyívóvonalat az óramutató járásával ellenkező irányban, amig nyívóvonalnak van még metszete az S halmazzal. Tehát keresni kell azt a legnagyobb lejtőszöget, amelynek a megfelelő nyívóvonalnak a metszete az S halmazzal nem üres. Vegyük észre, hogy ilyen legnagyobb szög az lesz, amelynél a nyívóvónal teljesen egybesik a $-x_1 + x_2 = 4$ egyenletű egyenessel.

Tehát a feladatnak végtelen sok optimális megoldása van – ezek közül az egyik a $(0, 4)$ csúcspontban található:

$$Q(0,4) \;=\; -0 - 4 - 22 \times 0 + 3 \times 4 + 3 \;=\; \frac{-6}{15} \;=\; -\frac{2}{5} \,.$$

A $-x_1 + x_2 = 4$ egyenletű egyenesen lévő végtelen távoli x pontban az optimális értéke a $Q(x)$ függvénnyek ugyanaz. Valóban, fejezzünk ki az x_2 változót a $-x_1 + x_2 = 4$ egyenletből:

$$x_2 = x_1 + 4$$

és ezt behelyettesítve a cél-függvénybe, a következőt kapjuk:

$$Q(x_1) \;=\; \frac{-x_1 - (x_1 + 4) - 2}{2x_1 + 3(x_1 + 4) + 3} \;=\; \frac{-x_1 - x_1 - 4 - 2}{2x_1 + 3x_1 + 12 + 3} \;=\; \frac{-2x_1 - 6}{5x_1 + 15} \,.$$

Ebből adódik, hogy

$$\lim_{x_1 \to \infty} Q(x_1) \;=\; \lim_{x_1 \to \infty} \frac{-2x_1 - 6}{5x_1 + 15} \;=\; -\frac{2}{5} \,.$$

Az alábbi hiperbolikus programozási feladattal demonstráljuk a (4) esetet, amikoris az x_1 és x_2 változók bármilyen nagy érteket felvehetnek a lehetséges megoldások S halmazán.

$$Q(x) \;=\; \frac{-x_1 - x_2 - 2}{2x_1 + 3x_2 + 3} \longrightarrow \max$$

a következő feltételek mellett

$$-x_1 + x_2 \leq 3$$

$$x_1 \geq 0, \;\; x_2 \geq 0$$

Vegyük észre, hogy az adott feladatban cél-függvény ugyanaz, mint az előző példában. Ami az S lehetséges halmazt illeti, egyetlen egy valtozás történt meg az előző példával szemben – a feltétel jobb oldalán 4 helyett 3 szerepel.

Mivel cél-függvény nem változott, nyilvánvaló hogy F fókusz pont ugyanaz, azaz $F = (-3, 1)$. Ezért a nyívóvonalt addig kell forgatni (az óramutató járásával ellenkező irányban) amig az nem lesz párhuzamos a $-x_1 + x_2 = 3$ egyenletű egyenessel. Vegyük észre, hogy ekkor a nyívóvonal metszi az S halmazt olyan $x = (\infty, \infty)$ pontban, amely a $-x_1 + x_2 = 3$ egyenletű egyenesen fekszik. Mas szavakkal, ebben az $x = (\infty, \infty)$ pontban teljesül $-x_1 + x_2 = 3$ egyenlet. Ezért,

$$x_2 = 3 + x_1$$

és

$$Q(x_1) \;=\; \frac{-x_1 - (x_1 + 3) - 2}{2x_1 + 3(x_1 + 3) + 3} \;=\; \frac{-x_1 - x_1 - 3 - 2}{2x_1 + 3x_1 + 9 + 3} \;=\; \frac{-2x_1 - 5}{5x_1 + 12}\,.$$

Ebből következik, hogy

$$\lim_{x_1 \to \infty} Q(x_1) \;=\; \lim_{x_1 \to \infty} \frac{-2x_1 - 5}{5x_1 + 12} \;=\; -\frac{2}{5}\,.$$

Tehát az optimum értéke $-2/5$, és az optimális megoldás az $-x_1 + x_2 = 3$ egyenletű egyenes végtelen távoli pontja.

A fejezet befejezéseként megemlítjük, hogy az itt tárgyalt grafikus technika minden olyan feladatra alkalmazható, amelyek felírhatók két független változóra vonatkozó egyenlőtlenségekkel. (Például egy három változót, egy egyenletet és egyenlőtlenségeket tartalmazó feladat átírható ilyen feladattá. Az egyenletből ki kell fejezni egy változót, majd ezzel a kifejezéssel kell helyettesíteni a változó minden előfordulását. Ilyen átíráskor ügyelni kell arra, hogy a kifejezett változó nemnegatív, így új feltételként fel kell venni a változó kifejezésének a nemnegativitását.) Megemlítjük még, hogy magasabb dimenziók esetében is lehet adni grafikus interpretációt. Így az n-dimenziós feladatnál a $Q(x)$ függvény számlálója és nevezője által meghatározott hipersíkok metszete lesz az $n-2$-dimenziós "fókusztengely", és ezen tengely körül "forognak" a $Q(x)$ függvény nívósíkjai.

2.3. Gyakorlatok

Oldjuk meg grafikus módszerrel az alábbi hiperbolikus programozási feladatokat.

1.:

$$Q(x) = \frac{P(x)}{D(x)} = \frac{10x_1 + 5x_2 + 2}{1x_1 + 4x_2 - 4} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{aligned} x_1 - 3x_2 &\geq -30 \\ x_1 + 2x_2 &\geq 20 \\ 2x_1 - x_2 &\leq 10 \end{aligned}$$

$$x_1 \geq 0, \quad x_2 \geq 0 \, .$$

2.:

$$Q(x) = \frac{P(x)}{D(x)} = \frac{2x_1 + 3x_2 + 3}{3x_1 + 5x_2 + 6} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{aligned} 8x_1 + 5x_2 &\leq 40 \\ 4x_1 + 9x_2 &\leq 36 \\ x_1 + 6x_2 &\geq 12 \\ 10x_1 + x_2 &\geq 10 \end{aligned}$$

$$x_1 \geq 0, \quad x_2 \geq 0 \, .$$

3.:

$$Q(x) = \frac{P(x)}{D(x)} = \frac{3x_1 - x_2 - 3}{x_1 + x_2 - 1} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{aligned} x_1 - x_2 &\geq 1 \\ x_1 + 2x_2 &\leq 14 \\ x_1 - 2x_2 &\leq 2 \end{aligned}$$

$$x_1 \geq 0, \quad x_2 \geq 0$$

4.:

$$Q(x) = \frac{P(x)}{D(x)} = \frac{x_1 + x_2 + 1}{2x_1 + 3x_2 + 8} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{aligned} -4x_1 + x_2 &\leq 1 \\ -x_1 - 4x_2 &\leq -4 \\ 4x_1 - x_2 &\leq 16 \end{aligned}$$

$$x_1 \geq 0, \quad x_2 \geq 0 \, .$$

5.:

$$Q(x) = \frac{P(x)}{D(x)} = \frac{2x_1 + x_2 - x_3 + 2}{x_1 + x_2 + 3x_3 + 1} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{array}{rcrcrcr} x_1 & + & 2x_2 & - & x_3 & = & 4 \\ x_1 & + & & + & 2x_3 & \leq & 22 \\ x_1 & + & x_2 & - & 2x_3 & \leq & 0 \\ x_1 & + & x_2 & & & \geq & 4 \end{array}$$

$$x_1 \geq 0, \;\; x_2 \geq 0 \; .$$

6.:

$$Q(x) \;=\; \frac{P(x)}{D(x)} \;=\; \frac{x_1 + x_2 + 1}{2x_1 + 3x_2 + 8} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{array}{rcrcr} x_1 & + & x_2 & \leq & 2 \\ -3x_1 & - & 2x_2 & \leq & -12 \\ 3x_1 & + & 8x_2 & \geq & 24 \end{array}$$

$$x_1 \geq 0, \;\; x_2 \geq 0 \; .$$

7.:

$$Q(x) \;=\; \frac{P(x)}{D(x)} \;=\; \frac{2x_1 - 2x_2}{5x_1 + 3x_2 - 16} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{array}{rcrcr} x_1 & + & x_2 & \leq & 8 \\ -4x_1 & + & 6x_2 & \leq & 24 \end{array}$$

$$x_1 \geq 0, \;\; x_2 \geq 0 \; .$$

3. Charnes–Cooper-transzformáció

A jelen alfejezetben egy olyan megoldási eljárással fogunk megismerkedni, amely segítségével tetszőleges hiperbolikus programozási feladathoz hozzá lehet rendelni egy neki megfelelő lineáris programozási feladatot és ezáltal a megoldandó hiperbolikus programozás feladat megoldását vissza lehet vezeteni a hozzárendelt lineáris programozási feladat megoldásához.
Ezt az eljárást A.Charnes és W.W.Cooper [**40**] publikálta 1962-ben.

3.1. Elmélet

Ebből a célból tekintsük az alábbi általános hiperbolikus programozási feladatot:

$$Q(x) = \frac{P(x)}{D(x)} = \frac{\sum_{j=1}^{n} p_j x_j + p_0}{\sum_{j=1}^{n} d_j x_j + d_0} \longrightarrow \max(\text{ or } \min) \tag{15}$$

a következő feltételek mellett

$$\begin{cases} \sum_{j=1}^{n} a_{ij} x_j \le b_i, & i = 1, 2, \ldots, m_1, \\ \sum_{j=1}^{n} a_{ij} x_j \ge b_i, & i = m_1 + 1, m_1 + 2, \ldots, m_2, \\ \sum_{j=1}^{n} a_{ij} x_j = b_i, & i = m_2 + 1, m_2 + 2, \ldots, m, \end{cases} \tag{16}$$

$$x_j \ge 0, \quad j = 1, 2, \ldots, n_1, \;\; n_1 \le n, \tag{17}$$

amelyről a továbbiakban hivatkozás nélkül mindig feltételezzük a következőket:

(1) a feladat S lehetséges halmaza korlátos és nem-üres, azaz $S \neq \emptyset$;
(2) $D(x) > 0, \;\; \forall x \in S$.

Vegyük észre, hogy a felsorolt feltételek biztosítják, hogy a(z) (15)-(17) feladat megoldható, és az optimális megoldása véges csúcspontban van. Valóban, mivel az S lehetséges halmaz korlátos és zárt, ezért az x_1 és x_2 változók csak korlátos értékeket vehetnek fel. Másrészt a $P(x)$ és $D(x)$ függvények lineárisak (azaz folytonosak), mégpedig $D(x) > 0, \;\; \forall x \in S$. Ebből következik, hogy nem üres, zárt, korlátos S halmazon a $Q(x)$ függvény felveszi véges értékű maximumát, illetve minimumát.

A Charnes–Cooper-transzformáció menetét követve vezessük be a következő új változókat:

$$t_j = \frac{x_j}{D(x)}, \;\; j = 1, 2, \ldots, n, \qquad t_0 = \frac{1}{D(x)},$$

ahol

$$D(x) = \sum_{j=1}^{n} d_j x_j + d_0. \tag{18}$$

Írjuk át az eredeti $Q(x)$ cél-függvényt a következő formába:

$$Q(x) \;=\; \frac{\sum_{j=1}^{n} p_j x_j + p_0}{\sum_{j=1}^{n} d_j x_j + d_0} \;=\; \frac{\sum_{j=1}^{n} p_j x_j + p_0}{D(x)} \;=\; \sum_{j=1}^{n} p_j \frac{x_j}{D(x)} + p_0 \frac{1}{D(x)}.$$

Az új $t_j,\ j = 0, 1, \ldots, n,$ változók használatával kapjuk az új cél-függvényt:

$$L(t) \;=\; \sum_{j=0}^{n} p_j t_j \longrightarrow \max(\text{ or } \min)\,. \tag{19}$$

Továbbá, mivel $D(x) > 0, \quad \forall x \in S,$ a(z) (16) és (17) feltételek mindkét oldalát oszthatjuk $D(x)$ -szel, és ezzel kaphatjuk az alábbi feltételeket:

$$\left.\begin{aligned} -b_i t_0 + \sum_{j=1}^{n} a_{ij} t_j &\le 0, \qquad i = 1, 2, \ldots, m_1, \\ -b_i t_0 + \sum_{j=1}^{n} a_{ij} t_j &\ge 0, \qquad i = m_1 + 1, m_1 + 2, \ldots, m_2, \\ -b_i t_0 + \sum_{j=1}^{n} a_{ij} t_j &= 0, \qquad i = m_2 + 1, m_2 + 2, \ldots, m, \end{aligned}\right\} \tag{20}$$

$$t_j \ge 0, \quad j = 0, 1, 2, \ldots, n_1, \;\; n_1 \le n, \tag{21}$$

Az új és az eredeti változók közötti kapcsolat csak akkor lesz teljes és hiánytalan, ha a(z) (18) kifejezést osztjuk $D(x)$ -szel, és az ilyen módon kapott

$$\sum_{j=0}^{n} d_j t_j = 1 \tag{22}$$

új feltételt hozzácsatoljuk az új feladat feltételrendszeréhez.

Az ilyen módon kapott (19)-(22) feladatot a továbbiakban a hiperbolikus programozási feladat *lineáris analógjának* fogjuk nevezni. Vegyünk észre, hogy a(z) (19)-(22) feladatban szereplő $L(t)$ cél-függvény lineáris, a feladat összes feltétele szintén lineáris. Tehát, maga a(z) (19)-(22) feladat is lineáris. Ezenkívül az eredeti HP feladatban van n változó és m feltétel, a lineáris analógban pedig szerepel $(n + 1)$ változó és $(m + 1)$ feltétel.

Joggal vetődik fel a kérdés, hogy az eredeti HP feladat és a lineáris analógja között mennyire szoros a kapcsolat, illetve mennyivel visz közelebb bennünket a lineáris analóg az eredeti HP feladat megoldásához.

Ezekre a kérdésekre a következő állítások adnak választ.

I.1. Lemma. *Ha* $t = (t_0, t_1, \ldots, t_n)$ *vektor a(z) (19)-(22) lineaáris analóg lehetséges megoldása, akkor* $t_0 > 0$.

I.1. Tétel. *Ha* $t^* = (t_0^*, t_1^*, \ldots, t_n^*)$ *vektor a(z) (19)-(22) lineáris analóg optimális megoldása, akkor az* $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ *vektor az eredeti (15)-(17) HP feladat optimális megoldása, megpedig*

$$x_j^* = \frac{t_j^*}{t_0^*}, \quad j = 1, 2, \ldots, n. \tag{23}$$

A következőkben egy numerikus példa megoldásával mutatjuk be a Charnes–Cooper-transzformáció gyakorlati végrehajtását.

3.2. Numerikus példa

Legyen a következő HP feladat:

$$Q(x) = \frac{8\,x_1 + 9\,x_2 + 4\,x_3 + 4}{2\,x_1 + 3\,x_2 + 2\,x_3 + 7} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{array}{rcrcrcr} 1\,x_1 & + & 1\,x_2 & + & 2\,x_3 & \leq & 3\,, \\ 2\,x_1 & + & 1\,x_2 & + & 4\,x_3 & \leq & 4\,, \\ 5\,x_1 & + & 3\,x_2 & + & 1\,x_3 & \leq & 15\,, \end{array}$$

$$x_j \geq 0, \quad j = 1, 2, 3.$$

A feladat optimális megoldása:

$$x^* = (1, 2, 0)^T, \quad P(x^*) = 30, \quad D(x^*) = 15, \quad Q(x^*) = 2\,.$$

A(z) (19)-(22) képleteknek megfelelően a a következő lineáris analóg megfelel az eredit HP feladatnak:

$$L(t) = 4\,t_0 + 8\,t_1 + 9\,t_2 + 4\,t_3 \longrightarrow \max$$

a következő feltételek mellett

$$\begin{array}{rcrcrcrcr} 7\,t_0 & + & 2\,t_1 & + & 3\,t_2 & + & 2\,t_3 & = & 1\,, \\ -3\,t_0 & + & 1\,t_1 & + & 1\,t_2 & + & 2\,t_3 & \leq & 0\,, \\ -4\,t_0 & + & 2\,t_1 & + & 1\,t_2 & + & 4\,t_3 & \leq & 0\,, \\ -15\,t_0 & + & 5\,t_1 & + & 3\,t_2 & + & 1\,t_3 & \leq & 0\,, \end{array}$$

$$t_j \geq 0, \quad j = 0, 1, 2, 3.$$

Ennek a lineáris programozási feladatnak az optimális megoldása:

$$t^* = (\frac{1}{15}, \frac{1}{15}, \frac{2}{15}, 0)^T, \quad L(t^*) = 2\,.$$

vagy

$$t_0^* = \frac{1}{15}\,,\ t_1^* = \frac{1}{15}\,,\ t_2^* = \frac{2}{15}\,,\ t_3^* = \frac{0}{15}\,.$$

Úgy, hogy a(z) (23) képleteknek megfelelően

$$x_1^* = \frac{t_1^*}{t_0^*} = \frac{\frac{1}{15}}{\frac{1}{15}} = 1, \quad x_2^* = \frac{t_2^*}{t_0^*} = \frac{\frac{2}{15}}{\frac{1}{15}} = 2, \quad x_3^* = \frac{t_3^*}{t_0^*} = \frac{\frac{0}{15}}{\frac{1}{15}} = 0,$$

Meg kell jegyeznünk, hogy korlátlan S lehetséges halmaz esetén előfordulhat, hogy a megoldandó HP feladat lineáris analógjának optimális megoldásában $t_0^* = 0$. Ilyenkor az eredeti HP feladat optimális megoldasa aszimptotikus, azaz x^* optimális megoldás tartalmaz végtelen értékű elemeket. Az aszimptotikus eset részletes leírását Gavurin, M.K. [**70**] cikkében találhatjuk.

Az alfejezet befejezése előtt még egy fontos megjegyzést kell tennünk.

A(z) I.1 tételben megfogalmazott, az eredeti HP feladat és a hozzá tartozó lineáris analóg közötti kapcsolat első nézetre rendkívülien hasznosnak és fontosnak látszik legalábbis azért, mert lehetővé teszi az eredeti megoldandó HP feladatot helyettesíteni speciális LP feladattal, és ezáltal nyílik lehetőség alkalmazni a jól ismert lineáris programozási matematikai apparátust.
A gyakorlatban azonban az adott megközelítés nem mindig hasznos és használható.

Nehézségek akkor merülnek fel, amikor egy speciális struktúrájú HP feladatról van szó. Például ha egy szállítási (lásd. V. fejezet) vagy hozzárendelési feladatot kell Charnes–Cooper-transzformáció használatával átalakítani lineáris alakba, akkor figyelembe kell vennünk, hogy az eredeti n ismeretlen változó helyett a lineáris analógban $(n + 1)$ változót kell kezelnünk, még az eredeti m feltétel helyett $(m + 1)$ -t fogunk kapni. Mindez annyira megváltoztatja a feladat eredeti struktúráját, hogy a feladatnak megfelelő speciális módszerek alkalmazása lehetlenné válik.

Ezenkívül a(z) (20) feltétel-rendszer jobb oldalán nincsen (szokásos értelemben) jobb oldali $b = (b_1, b_2, \ldots, b_m)$ vektor. E helyett szerepel m elemes $0 = (0, 0, \ldots, 0)$ nulla-vektor. Az utóbbi azt jelenti, hogy a lineáris programozási dualitás elmélete ebben az esetben nem használható.

Szóval speciális HP feladatok esetén a Charnes–Cooper-transzformáció nem könnyíti a feladat megoldását, sőt néha megnehezíti azt. Mindez szükségessé teszi a HP feladat közvetlen tanulmanyozását az LP feladatra történő visszavezetése nélkül.

3.3. Gyakorlatok

További gyakorláshoz önállóan konstruáljunk lineáris analógokat az alábbi HP feladatok számára:

3.3.1.:

$$Q(x) = \frac{2\,x_1 + x_2 - 2}{x_1 + x_2 + 1} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{array}{rcrcl} x_1 & + & x_2 & \leq & 3\,, \\ x_1 & - & x_2 & \leq & 2\,, \end{array}$$

$$x_j \geq 0, \;\; j = 1, 2.$$

3.3.2.:

$$Q(x) = \frac{2\,x_1 + 6\,x_2 + 2\,x_3 - 3}{5\,x_1 + 1\,x_2 + 4\,x_3 + 1} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{array}{rcrcrcr} 3\,x_1 & - & 2\,x_2 & + & 1\,x_3 & \leq & 135\,, \\ 1\,x_1 & + & 3\,x_2 & + & 3\,x_3 & \geq & 45\,, \\ 8\,x_1 & - & 5\,x_2 & + & 1\,x_3 & = & 150\,, \end{array}$$

$$x_j \geq 0, \;\; j = 1, 2, 3.$$

3.3.3.:

$$Q(x) = \frac{5x_1 - 3x_2 + 2}{4x_1 + 1x_2 - 2} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{array}{rcrcl} x_1 & + & 2x_2 & \geq & 4\,, \\ x_1 & + & 3x_2 & \leq & 6\,, \end{array}$$

$$x_1 \geq 0, \;\; x_2 \geq 0$$

4. Dinkelbach-algoritmus

Az egyik legáltalánosabb és leggyakrabban használt stratégia a tört alakú célfüggvényes optimalizálási feladatok megoldásakor (nem csak linearis $P(x)$ és $D(x)$ esetén) a W.Dinkelbach [**56**] által bevezetett parametrikus módszer. A hiperbolikus programozási feladat esetén a módszer lényege abban áll, hogy az eredeti megoldandó HP feladat megoldását visszavezetjük a lineáris programozási feladatok sorozatának megoldásához.

4.1. Elmélet

Tekintsük a következő hiperbolikus programozási feladatot:

$$Q(x) = \frac{P(x)}{D(x)} = \frac{\sum_{j=1}^{n} p_j x_j + p_0}{\sum_{j=1}^{n} d_j x_j + d_0}, \longrightarrow \max(\text{ or } \min) \tag{24}$$

a következő feltételek mellett

$$\begin{cases} \sum_{j=1}^{n} a_{ij} x_j \leq b_i, & i = 1, 2, \ldots, m_1, \\ \sum_{j=1}^{n} a_{ij} x_j \geq b_i, & i = m_1 + 1, m_1 + 2, \ldots, m_2, \\ \sum_{j=1}^{n} a_{ij} x_j = b_i, & i = m_2 + 1, m_2 + 2, \ldots, m, \end{cases} \tag{25}$$

$$x_j \geq 0, \quad j = 1, 2, \ldots, n_1, \tag{26}$$

és vezessük be a következő függvényt:

$$F(\lambda) = \max_{x \in S} \{P(x) - \lambda D(x)\}, \ \ \lambda \in R,$$

ahol S a szokásos módon jelöli a(z) (25)-(26) feltételek által meghatározott lehetséges halmazt, és λ egy paraméter.

A Dinkelbach-módszer alapötletének elméleti hátttereként a következő állítás szolgál:

I.2. Tétel. *Egy x^* vektor a(z) (24)-(26) HP feladat optimális megoldása akkor és csak akkor, ha teljesül az*

$$F(\lambda^*) = \max_{x \in S} \{P(x) - \lambda^* D(x)\} \ = \ 0 \tag{27}$$

egyenlőség, ahol

$$\lambda^* = \frac{P(x^*)}{D(x^*)}. \tag{28}$$

A fenti tétel nemcsak választ ad arra a kédésre, hogy egy lehetséges x vektor vajon optimális-e, hanem alapjául szolgál az optimális megoldáshoz vezető algoritmusnak is.

Valóban, mivel $D(x) > 0, \ \forall x \in S$, ezért

$$\frac{\partial F(\lambda)}{\partial \lambda} \ = \ -D(x) \ < \ 0.$$

Az utóbbi azt jelenti, hogy az $F(\lambda)$-ra adott (27) kifejezés szigorúan csökkenő λ-tól függő függvény.

Ezért a Dinkelbach-módszer megvalósítható a következő lépésekben:

0.Lépés: : Határozzunk meg egy $x^{(0)}$ tetszőleges induló lehetséges megoldást, azaz $x^{(0)} \in S$. Legyen $k := 1$ és $\lambda^{(1)} = P(x^{(0)})/D(x^{(0)})$.

1.Lépés: : Oldjuk meg a következő lineáris programozási feladatot:

$$\max_{x \in S} \{P(x) - \lambda^{(k)} D(x)\}$$

Jelöljük $x^{(k)}$-val a kapott megoldást, azaz

$$x^{(k)} := \arg\max_{x \in S} \{P(x) - \lambda^{(k)} D(x)\}.$$

2.Lépés: : Ha $F(\lambda^{(k)}) = P(x^{(k)}) - \lambda^{(k)} D(x^{(k)}) = 0$, akkor az $x^* = x^{(k)}$ vektor a keresett optimális megoldás. Stop.
Különben

3.Lépés: : Legyen $\lambda^{(k+1)} := P(x^{(k)})/D(x^{(k)})$. Legyen $k := k + 1$. Vissza az 1.Lépéshez.

4.2. Numerikus példa

Az alábbi numerikus példa illusztrálja a módszer működését. Tekintsük a következő hiperbolikus programozási feladatot:

$$Q(x) \;=\; \frac{P(x)}{D(x)} \;=\; \frac{x_1 + x_2 + 5}{3x_1 + 2x_2 + 15} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{array}{rcrcr} 3x_1 & + & x_2 & \leq & 6\,, \\ 3x_1 & + & 4x_2 & \leq & 12\,, \end{array} \tag{29}$$
$$x_1 \geq 0, \;\; x_2 \geq 0$$

0. Lépés: Mivel $x = (0, 0)$ vektor kielégíti minden feltételét a megoldandó feladatnak, ezért választhatjuk ezt a vektort az induló pontnak. Szóval, legyen $x^{(0)} = (0, 0)$. Tehát a kiválasztott $x^{(0)} = (0, 0)$ pont esetén

$$\lambda^{(1)} := \frac{P(x^{(0)})}{D(x^{(0)})} \;=\; \frac{5}{15} \;=\; \frac{1}{3},$$

1. Lépés: Most a következő lineáris programozási feladatot kell megoldanunk

$$P(x) - \lambda^{(1)} D(x) = P(x) - \frac{1}{3} D(x) = \frac{1}{3}\, x_2 \longrightarrow \max$$

a(z) (29) feltételek mellett. Ennek a feladatnak optimális megoldása

$$x^{(1)} \;=\; (0,3),$$

úgyhogy

$$F(\lambda^{(1)}) \;=\; 1\,.$$

2. Lépés: Mivel $F(\lambda^{(1)}) \neq 0$, végre kell hajtanunk a következő lépést:
3. Lépés: Most ki kell számolnunk az új $\lambda^{(2)}$-t:

$$\lambda^{(2)} := \frac{P(x^{(1)})}{D(x^{(1)})} \;=\; \frac{1\times 3+5}{2\times 3+15} \;=\; \frac{8}{21},$$

és növelnünk kell a k értékét, azaz $k := k+1 = 2$. Vissza az **1. Lépés**hez.
1. Lépés: Oldjuk meg a következő lineáris programozási feladatot:

$$\begin{aligned} P(x) \;-\; & \lambda^{(2)} D(x) \;= \\ =\; & (1-\frac{8}{21}\times 3)x_1 + (1-\frac{8}{21}\times 2)x_2 + (5-\frac{8}{21}\times 15) = \\ =\; & -\frac{1}{7}\,x_1 + \frac{5}{21}\,x_2 - \frac{5}{7} \longrightarrow \max \end{aligned}$$

a(z) (29) feltételek mellett.

Megoldva ezt a feladatot a következőket kapjuk:

$$x^{(2)} = (0,3) \quad \text{és} \quad F(\lambda^{(2)}) \;=\; 0\,.$$

2. Lépés: Mivel $F(\lambda^{(2)}) = 0$, ezért az $x^* = x^{(2)}$ vektor a keresett optimális megoldása a megoldandó hiperbolikus programozási feladatnak; Stop;

Szóval a fenti hiperbolikus programozási feladat optimális megoldása

$$x^* = (0,3), \quad Q(x^*) = 8/21.$$

4.3. Gyakorlatok

Oldjuk meg Dinkelbach-módszerrel az alábbi hiprbolikus programozási feladatokat. A Dinkelbach-algoritmus végrehajtása során keletkező lineáris programozási programozási feladatokat oldjuk meg alkalmas szoftver eszközökkel, pl.: LinDo, LinGo, Excel/Solver vagy WinGULF.

4.3.1.:

$$Q(x) = \frac{2x_1 + x_2 + 15}{3x_1 + x_2 + 25} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{aligned} 1x_1 \;+\; & x_2 \;\leq\; 15\,, \\ 3x_1 \;+\; & 4x_2 \;\leq\; 65\,, \end{aligned}$$

$$x_1 \geq 0, \;\; x_2 \geq 0$$

4.3.2.:

$$Q(x) = \frac{3x_1 + x_2 - 5}{7x_1 + 2x_2 + 15} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{array}{rcrcr} -3x_1 & + & x_2 & \geq & 6\,, \\ 3x_1 & + & 5x_2 & \leq & 15\,, \end{array}$$

$$x_1 \geq 0,\ \ x_2 \geq 0$$

4.3.3.:

$$Q(x) = \frac{5x_1 - 3x_2 + 2}{4x_1 + 1x_2 + 5} \longrightarrow \min$$

a következő feltételek mellett

$$\begin{array}{rcrcr} x_1 & + & 2x_2 & \leq & 4\,, \\ x_1 & + & 3x_2 & \geq & 6\,, \end{array}$$

$$x_1 \geq 0,\ \ x_2 \geq 0$$

4.3.4.:

$$Q(x) = \frac{5x_1 - 3x_2 + 2}{4x_1 + 1x_2 - 2} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{array}{rcrcr} x_1 & + & 2x_2 & \geq & 4\,, \\ x_1 & + & 3x_2 & \leq & 6\,, \end{array}$$

$$x_1 \geq 0,\ \ x_2 \geq 0$$

4.3.5.:

$$Q(x) = \frac{x_1 + 3x_2 + 2}{x_1 + 2x_2 + 5} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{array}{rcrcr} 3x_1 & + & x_2 & \geq & 14\,, \\ x_1 & + & 3x_2 & \leq & 26\,, \end{array}$$

$$x_1 \geq 0,\ \ x_2 \geq 0$$

II. Fejezet
Szimplex módszer

Az 1940-es években George Dantzig amerikai matematikus által kifejlesztett, eredetileg lineáris programozási feladat megoldására szolgáló *szimplex módszert* később, 1960.-ban Martos Béla magyar matematikus adaptálta a hiperbolikus programozási feladatra.

Ebben a fejezetben a szimplex módszer hiperbolikus programozási változatával fogunk foglalkozni. A fejezet a következőképpen épül fel. Elsőként a módszer elméleti hátterével foglalkozunk, majd áttérünk az algoritmus végrehajtási kérdéseire. Ezt követően néhány numerikus példa alapján bemutatjuk a módszer működését.

Legyen a következő kanonikus alakú hiperbolikus programozási feladat:

$$Q(x) = \frac{P(x)}{D(x)} = \frac{\sum_{j=1}^{n} p_j x_j + p_0}{\sum_{j=1}^{n} d_j x_j + d_0} \to \max, \tag{30}$$

a következő feltételek mellett

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1, 2, \ldots, m, \tag{31}$$

$$x_j \geq 0, \quad j = 1, 2, \ldots, n\ . \tag{32}$$

Itt és mindenütt a továbbiakban ebben a fejezetben feltételezzük, hogy

(1) az S lehetséges halmaz nem üres és korlátos ,
(2)

$$D(x) > 0 \quad \forall x \in S \tag{33}$$

1. Fő definíciók és tételek

II.1. Definíció. *A(z) (30)-(32) maximalizálási hiperbolikus programozási feladatot* ***megoldható****nak fogjuk nevezni, ha*

- *a(z) (31)-(32) feltételek által meghatározott S lehetséges halmaz nem üres, azaz létezik legalább egy olyan $x = (x_1, x_1, \ldots, x_n)$ vektor, amely kielégíti a feladat (31)-(32) feltételeit, és*
- *a $Q(x)$ cél-függvény az S lehetséges halmazon felülről korlátos.*

Egyébként a(z) (30)-(32) feladatot ***megoldhatatlan****nak fogjuk nevezni.*

Tekintsük a következő lineáris egyenletrendszert:

$$\sum_{j=1}^{n} A_j x_j = b,$$

ahol

$$A_j = \begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{pmatrix},\ j = 1, 2, \ldots, n, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}, \quad \text{és} \ \ m \leq n.$$

II.2. Definíció. *Azt fogjuk mondani, hogy az A_j vektorokból álló*

$$B = \{A_{s_1}, A_{s_2}, \ldots, A_{s_m}\}$$

vektorrendszer egy ***bázis****t alkot, ha $A_{s_1}, A_{s_2}, \ldots, A_{s_m}$ vektorok lineárisan függetlenek, azaz a B vektorrendszer lineárisan független.*

Tegyük fel, hogy az adott $B = \{A_{s_1}, A_{s_2}, \ldots, A_{s_m}\}$ vektorrendszer bázis. Jelölje J_B a B bázishoz tartozó A_j vektorok indexének halmazát, azaz

$$J_B = \{s_1, s_2, \ldots, s_m\}.$$

Ha $J = \{1, 2, \ldots, n\}$, akkor a $J_N = J \backslash J_B$ indexhalmaz jelöli azon A_j vektorok indexének halmazát, amelyek nem szerepelnek az adott B bázisban.

II.3. Definíció. *Az adott $x = (x_1, x_2, \ldots, x_n)^T$ vektort* ***bázismegoldás****nak (vagy* ***bázisvektor****nak) fogjuk nevezni, ha az x vektor kielégíti a*

$$\sum_{j \in J_B} A_j x_j = b$$

rendszert, és

$$x_j = 0, \ \ \forall j \in J_N.$$

Azokat az x_j változókat, amelyek indexe a J_B indexhalmazban vannak **bázisváltozók**nak szokás nevezni.

Ha egy x_j változó nem szerepel a bázisban, akkor azt **nem-bázisváltozó**nak fogjuk nevezni, azaz ha x_j nem-bázisváltozó, akkor $j \in J_N$.

Extermális pont (csúcspont) fogalma:

II.4. Definíció. *Az S konvex halmaz x pontját **extremális pont**nak (vagy **csúcspont**nak) nevezzük, ha az S halmazban nem léteznek olyan x' és x'' pontok, amelyeknek az x pont lineáris kombonációja, azaz $x = \lambda x' + (1-\lambda)x''$, ahol $0 < \lambda < 1$,*

Szomszéd (szomszédos pont) fogalma:

II.5. Definíció. *Ha adott egy $Ax = b$ lineáris egyenletrendszer n változóval és m feltétellel, akkor ennek az egyenletrendszernek két tetszőleges bázismegoldását **szomszédos** pontoknak nevezzük, ha ezeknek a bázismegoldásoknak megfelelő bázisok egymástól egyetlen egy vektorban különböznek, azaz $(m-1)$ vektor ugyanaz.*

Az extremális pontok vagy csúcspontok nagyon fontos szerepet játszanak a szimplex módszerben.

A csúcspontok és bázismegoldások közötti kapcsolatot a következő tétel határozza meg:

II.1. Tétel. *Az $Ax = b$ lineáris egyenletrendszer által meghatározott konvex halmaz egy x pontja akkor és csak akkor csúcspont, ha x vektor ennek a rendszernek bázismegoldása.*

Más szavakkal minden csúcspontnak megfelel legalább egy bázismegoldás (és ezek szerint legalább egy bázis).

II.6. Definíció. *Azt mondjuk, hogy az x bázismegoldás **degenerált**, ha a bázisváltozók között legalább egy változó nulla értékű, azaz $\exists j : \ j \in J_B$, és $x_j = 0$. Abban az esetben, ha $x_j \neq 0$, $\forall j \in J_B$, akkor az x bázismegoldást **nem-degenerált**nak nevezzük.*

A nem-degenerált bázismegoldás fogalma nagyon fontos szerepet játszik a szimplex módszerben, mert a degenerált csúcspontnak általános esetben több bázis és ezért több bázismegoldás felel meg.

II.7. Definíció. *A(z) (31) rendszer $x = (x_1, x_2, \ldots, x_n)^T$ bázismegoldását a(z) (30)-(32) feladat **lehetséges bázismegoldásának** nevezzük, ha minden x_j eleme nem-negatív, azaz $x_j \geq 0 \ \ j = 1, 2, \ldots, n,$.*

II.8. Definíció. *A(z) (30)-(32) kanonikus hiperbolikus programozási feladatot **normálisnak** vagy **normalizáltnak** fogjuk nevezni, ha a(z) (31)*

egyenletrendszer jobb oldali $b = (b_1, b_2, \ldots, b_m)^T$ *vektorának összes* b_i *eleme nem-negatív, azaz*

$$b_i \geq 0, \quad i = 1, 2, , , \ldots, m.$$

A szimplex módszernek a hiperbolikus programozási feladat megoldására való alkalmazhatósága a következő tételen alapszik:

II.2. Tétel. (Monotonitás) *Az* S *lehetséges halmazhoz tartozó tetszőleges egyenes szakaszon a* $Q(x)$ *cél-függvény monoton.*

Bizonyítás. Azzal kezdjük bizonyítást, hogy válasszuk az S lehetséges halmazon két tetszőleges x' és x'' pontot, azaz $x' \in S$ és $x'' \in S$.
Tekintsük a $Q(x)$ cél-függvény viselkedését az $x'x''$ szakaszon. Válassszuk ezen a szakaszon egy tetszőleges x pontot, azaz $x = \lambda x' + (1-\lambda)x''$, ahol $0 \leq \lambda \leq 1$. Nyilvánvaló, hogy

$$Q(\lambda) = \frac{P(\lambda x' + (1-\lambda)x'')}{D(\lambda x' + (1-\lambda)x'')} = \ldots = \frac{\lambda P(x') + (1-\lambda)P(x'')}{\lambda D(x') + (1-\lambda)D(x'')}$$

és ennek megfelelően

$$\frac{dQ(\lambda)}{d\lambda} = \frac{P(x')D(x'') - P(x'')D(x')}{(\lambda D(x') + (1-\lambda)D(x''))^2}. \tag{34}$$

Mivel

$$(\lambda D(x') + (1-\lambda)D(x''))^2 > 0$$

a(z) (34) azt jelenti, hogy $x'x''$ szakaszon $Q(x)$ cél-függvény

- növekszik, ha $P(x')D(x'') - P(x'')D(x') > 0$,
- csökken, ha $P(x')D(x'') - P(x'')D(x') < 0$,
- nem változik, ha $P(x')D(x'') - P(x'')D(x') = 0$.

Igy a tétel bizonyítva van. $\diamondsuit$

Mivel S lehetséges halmaz konvex, ezért a(z) II.2. tételből következik, hogy

II.3. Tétel. *Ha (30)-(32) hiperbolikus programozás feladat* S *lehetséges halmaza korlátos, akkor* $Q(x)$ *cél-függvény a maximális értékét az* S *halmazon az* S *halmaz csúcspontjában veszi fel.*

II.1. Megjegyzés. *A(z) II.3. tétel csak abban az esetben érvényes, ha az* S *lehetséges halmaz korlátos. Korlátlan* S *halmazban előfordulhat az úgynevezett aszimptotikus eset (lásd I. fejezet 2. alfejezet)*

2. Optimalitási kritérium

Tegyük fel, hogy a(z) (30)-(32) hiperbolikus programozási feladat normális, azaz $b_i \geq 0, \quad i = 1, 2, \ldots, m$. Továbbá feltételezzük, hogy az $x = (x_1, x_2, \ldots, x_n)^T$ vektor a(z) (30)-(32) feladat nem-degenerált lehetséges bázismegoldása, és ennek a bázismegoldásnak megfelel a $B = (A_{s_1}, A_{s_2}, \ldots, A_{s_m})$ bázis. Ez azt jelenti, hogy

$$x_j \begin{cases} > 0, & j \in J_B = \{s_1, s_2, \ldots, s_m\}, \\ = 0, & j \in J_N = J \setminus J_B, \end{cases}$$

ahol $J = \{1, 2, , \ldots, n\}$. Ezeknek a feltételezéseknek megfelelően

$$\sum_{j=1}^{n} A_j x_j = \sum_{j \in J_B} A_j x_j + \sum_{j \in J_N} A_j x_j = \sum_{j \in J_B} A_j x_j + 0 = \sum_{i=1}^{m} A_{s_i} x_{s_i}.$$

Mivel az x vektor bázismegoldás, ezért

$$\sum_{i=1}^{m} A_{s_i} x_{s_i} = b. \tag{35}$$

A szimplex módszer eljárásának megfelelően válasszunk egy A_j nem-bázis vektort (azaz $j \in J_N$), és vezessük be ezt a vektort a bázisba. Mivel ilyenkor a bázis változik, ezért változnak a benne levő bázisváltozók, és az x_j új bázisváltozó új értékét kap (eredetileg nulla volt). Éppen ezek miatt a bázisban történő változások miatt a következő új jelölésekre van szükségünk:

– θ jelölje az x_j új bázisváltozó értékét, és
– $x_j(\theta)$ jelölje a bázisban már meglévő régi bázisváltozók új értékét.

Az új jelölések használatával a(z) (35) képlet átírható a következő formába:

$$\sum_{i=1}^{m} A_{s_i} x_{s_i}(\theta) + A_j \theta = b. \tag{36}$$

Továbbá, mivel az $A_{s_1}, A_{s_2}, \ldots, A_{s_m}$ bázisvektorok lineárisan független vektorrendszert alkotnak, ezért az A_j vektor ebben a rendszerben előállítható lineáris kombinációként:

$$A_j = \sum_{i=1}^{m} A_{s_i} x_{ij}. \tag{37}$$

Helyettesítsük a(z) (36) képletben az A_j vektort a(z) (37) egyenlet jobb oldalában szereplő kifejezéssel:

$$\sum_{i=1}^{m} A_{s_i} x_{s_i}(\theta) + \theta \sum_{i=1}^{m} A_{s_i} x_{ij} = b. \tag{38}$$

Vegyük figyelembe, hogy a(z) (35) és (38) egyenletek jobb oldalán szereplő kifejezések egyformák, ezért

$$\sum_{i=1}^{m} A_{s_i} x_{s_i}(\theta) + \theta \sum_{i=1}^{m} A_{s_i} x_{ij} = \sum_{i=1}^{m} A_{s_i} x_{s_i}$$

vagy másképpen

$$\sum_{i=1}^{m} A_{s_i} x_{s_i}(\theta) = \sum_{i=1}^{m} A_{s_i} (x_{s_i} - \theta x_{ij}) \ .$$

Mivel az $A_{s_1}, A_{s_2}, \ldots, A_{s_m}$ vektorok lineárisan függetlenek, ezért

$$x_{s_i}(\theta) = x_{s_i} - \theta x_{ij}, \quad i = 1, 2, \ldots, m. \tag{39}$$

A(z) (39) képlet használata az $x(\theta)$ új bázismegoldás előállításához biztosítja a(z) (31) feltételrendszer kielégítését. Azonban a(z) (39) képlet használata nem biztosítja az $x(\theta)$ vektor elemeinek nem-negativitását (32). Tehát nem biztos, hogy $x(\theta)$ lehetséges bázismegoldása lesz a feladatnak. Éppen ezért olyan θ-ra van szükségünk, amely mellett teljesülnek az

$$x_{s_i}(\theta) \geq 0, \quad i = 1, 2, \ldots, m,$$

vagy a(z) (39) képlet használatával

$$x_{s_i} - \theta x_{ij} \geq 0, \quad i = 1, 2, \ldots, m.$$

nem-negatívitási feltételek.

Nyilvánvaló, az utóbbit át lehet írni a következő formában:

$$\theta \leq \frac{x_{s_i}}{x_{ij}}, \quad \text{olyan } i \text{ index esetén, hogy } x_{ij} > 0,$$

$$\theta \geq \frac{x_{s_i}}{x_{ij}}, \quad \text{olyan } i \text{ index esetén, hogy } x_{ij} < 0,$$

vagy ugyanaz, csak kompaktabb formában:

$$\max_{x_{ij}<0} \frac{x_{s_i}}{x_{ij}} \leq \theta \leq \min_{x_{ij}>0} \frac{x_{s_i}}{x_{ij}} \tag{40}$$

Mivel θ-val jelöltük az x_j új bázisváltozó új értékét, ezért csak nemnegatív θ-t választhatunk, más szavakkal a(z) (40) feltétel helyett a

$$0 \leq \theta \leq \min_{x_{ij}>0} \frac{x_{s_i}}{x_{ij}}. \tag{41}$$

feltételt kell használnunk a θ érték kiválasztásához.

Továbbá θ értéke nem lehet nulla azért, mert ilyenkor az új bázisban csak $(m-1)$ vektor marad, és a(z) II.2. definició szerint ez már nem lesz bázis. Nekünk pedig új bázisra lenne szükségünk.

Ugyanilyen ok miatt nem választhatjuk θ értékéül a(z) (41) tartomány belső pontját, mert ebben az esetben az új vektorrendszerben $(m+1)$ darab A_j lenne. Ezért marad a következő egyetlen egy elfogadható θ érték:

$$\theta = \min_{x_{ij}>0} \frac{x_{s_i}}{x_{ij}}. \tag{42}$$

A(z) (42) képletet *minimális fajlagos vizsgálatnak* (az angol nyelvű szakirodalomban *minimum ratio test*) szokták nevezni.

II.2. Megjegyzés. *Előfordulhat, hogy a kiválasztott A_j vektor számára nincsen egyetlen egy olyan i index, amely melett $x_{ij} > 0$, és ennek következtében a(z) (40) tartományban véges felső korlát nem létezik. Ilyenkor θ értéke nem véges. Részletesebben ezt a esetet a 3. alfejezetben fogjuk elemezni.*
Egy másik "rossz" eset fordulhat elő, ha a(z) (42) vizsgálat eredményéül több ilyen i indexet kapunk (lasd. 8.2. alfejeztet).

Mivel már tudjuk, hogyan kell kiválasztani a θ értéket, tegyük fel hogy

$$\min_{x_{ij}>0} \frac{x_{s_i}}{x_{ij}} = \frac{x_r}{x_{rj}}.$$

Az utóbbi azt jelenti, hogy az új bázisban

$$x_r(\theta) = 0, \quad x_j = \theta$$

és az A_j vektor az új bázisban az A_r vektor helyébe kerül. Tehát az eredeti

$$B = \{A_{s_1}, A_{s_2}, \ldots, A_r, \ldots, A_{s_m}\}$$

bázis helyett az új

$$B' = \{A_{s_1}, A_{s_2}, \ldots, A_j, \ldots, A_{s_m}\}$$

bázist kapjuk.

Most pedig ki kell számolnunk a $Q(x)$ cél-függvény új értékét (azaz az $x(\theta)$ pontban):

$$Q(x(\theta)) = \frac{P(x(\theta))}{D(x(\theta))} = \frac{\sum_{j=1}^{n} p_j x_j(\theta) + p_0}{\sum_{j=1}^{n} d_j x_j(\theta) + d_0} = \frac{\sum_{i=1}^{m} p_{s_i} x_{s_i}(\theta) + p_j\theta + p_0}{\sum_{i=1}^{m} d_{s_i} x_{s_i}(\theta) + d_j\theta + d_0} =$$

$$= \frac{\sum_{i=1}^{m} p_{s_i}(x_{s_i} - \theta x_{ij}) + p_j\theta + p_0}{\sum_{i=1}^{m} d_{s_i}(x_{s_i} - \theta x_{ij}) + d_j\theta + d_0} =$$

$$= \frac{\sum_{i=1}^{m} p_{s_i} x_{s_i} + p_0 - \theta(\sum_{i=1}^{m} p_{s_i} x_{ij} - p_j)}{\sum_{i=1}^{m} d_{s_i} x_{s_i} + d_0 - \theta(\sum_{i=1}^{m} d_{s_i} x_{ij} - d_j)} = \frac{P(x) - \theta\Delta'_j}{D(x) - \theta\Delta''_j},$$

ahol

$$\Delta'_j = \sum_{i=1}^{m} p_{s_i} x_{ij} - p_j, \quad \Delta''_j = \sum_{i=1}^{m} d_{s_i} x_{ij} - d_j.$$

Amennyiben már kiderítettük cél-függvény új értékét, képesek vagyunk megbecsülni a $Q(x(\theta))$ és $Q(x)$ közötti különbséget:

$$Q(x(\theta)) - Q(x) = \frac{P(x) - \theta\Delta'_j}{D(x) - \theta\Delta''_j} - \frac{P(x)}{D(x)} = \ldots = \frac{-\theta\Delta_j(x)}{D(x(\theta))}, \tag{43}$$

ahol

$$\Delta_j(x) = \Delta'_j - Q(x)\Delta''_j = \begin{vmatrix} \Delta'_j & Q(x) \\ \Delta''_j & 1 \end{vmatrix}.$$

A(z) (43) képlet lehetővé teszi választ adni arra a kérdésre, hogy vajon érdemes volt-e bevezetni a bázisba az A_j vektort. Tényleg, mivel $\theta > 0$ (korábban a(z) 35. oldalon feltételeztük, hogy az x lehetséges bázismegoldás nem-degenerált, azaz minden báziseleme nem-nulla értékű) és $D(x(\theta)) > 0$ ($D(x) > 0$, $\forall x \in S$), ezért az A_j vektor a bázisba való bevezetése (és ezáltal az x lehetséges csúcspontból az $x(\theta)$ csúcspontba való átmenet) miatt a $Q(x)$ cél-függvény értéke csökken vagy növekszik attól függően, hogy milyen előjelű a $\Delta_j(x)$ determináns. Ha $\Delta_j(x) < 0$, akkor a $Q(x)$ cél-függvény értéke növekszik, ha $\Delta_j(x) > 0$, akkor $Q(x)$ csökken. Nyilvánvaló, hogy ha $\Delta_j(x) = 0$, a $Q(x)$ cél-függvény értéke változatlan marad.

Eredményeinket a következő, az "**Optimalitási kritérium**"nak nevezett tételként összegezhetjük

II.4. Tétel (Optimalitási kritérium). *Egy adott x lehetséges bázismegoldás akkor és csak akkor optimális, ha* $\Delta_j(x) \geq 0, \;\; j = 1, 2, \ldots, n.$

Nyilvánvaló, ha $d_j = 0, \; j = 1, 2, \ldots, n$ és $d_0 = 1$, akkor $\Delta_j(x) = \Delta'_j, \; j = 1, 2, \ldots, n$ és a(z) II.4. Tételből kapjuk a lineáris programozási szimplex módszer optimalitási kritériumát.

3. A szimplex módszer általános sémája

Ebben az alfejezetben azt mutatjuk meg, hogyan kell használni a szimplex módszert a hiperbolikus programozási feladat megoldására.

Itt és mindenütt a továbbiakban csak maximalizálási hiperbolikus programozási feladattal foglalkozunk, mert a minimalizálási feladat megoldása megfelelő maximalizálási feladatból kapható (lasd. (13)).

A szimplex módszer menete a következő

(1) Ha a megoldandó hiperbolikus programozási feladat nem kanonikus és nem normalizált alakú, akkor a feladatot át kell alakítani (lásd. 1.4. fejezet) úgy, hogy kanonikus és normál alakú legyen.

(2) Ha van lehetőség, keressünk egy induló lehetséges bázismegoldást. Ez lehet nagyon egyszerű, ha eredetileg a feladatban csak "$\leq$" relációjú feltételek szerepelnek, és a jobb oldali b vektorban minden elem nem-negatív. Ilyenkor az átalakításkor bevezetett az ún. *mesterséges* (lasd. 7. oldal) s_i változókat használhatjuk bázisváltozókként.
Ha az induló lehetséges bázismegoldás "manuális" előállítása nem könnyű feladat, akkor megfelelő módszert kell használnunk (lásd. 6.1. és 6.2. alfejezeteket).

(3) Ha az összes nem-bázis $x_j, \; \forall j \in J_N,$ változókhoz tartozó $\Delta_j(x)$ determinánsok nem-negatív értékűek, azaz

$$\text{ha} \quad \Delta_j(x) \geq 0, \; \forall j \in J_N,$$

akkor az aktuális lehetséges bázismegoldás optimális.
Ha van legalább egy negatív értékű determináns $\Delta_j(x)$, azaz

$$\exists j : \;\; \Delta_j(x) < 0, \; j \in J_N,$$

akkor a negatív $\Delta_j(x)$ determinánssal rendelkező nem-bázis változókból ki kell választani egyet (a megfelelő szabályokat később fogjuk tárgyalni, lásd 8.1. alfejezet). A kiválsztott nem-bázis változót *bevezetendő változónak* szokták nevezni, és a hozzá tartozó A_j vektort *bevezetendő vektornak* nevezik (angolul - "*entering variable*" és "*entering column-vector*").

(4) A kiválsztott változót és a megfelelő A_j vektort be kell vezetni a bázisba, és az új bázisban ki kell számolni az új $\Delta_j(x)$ determinánsokat. Majd vissza a 3. lépéshez.

Most pedig vizsgáljuk meg részletesseben a 3. lépést. A korábban megtett feltételezésünk szerint x nem-degenerált lehetséges bázismegoldás, amelynek megfelel a

$$B = (A_{s_1}, A_{s_2}, \ldots, A_{s_m})$$

bázis. Az előző alfejezetnek megfelelően jelölje J_B azon A_j vektorok j indexének a halmazát, amelyek a bázisban vannak, azaz

$$J_B = \{s_1, s_2, \ldots, s_m\}$$

Ha $J = \{1, 2, \ldots, n\}$, akkor $J_N = J \setminus J_B$ jelöli a nem-bázis vektorok index-halmazát.

Az x lehetséges bázismegoldás az optimalitásra való vizsgálata azzal kezdődik, hogy ki kell számolnunk a következő értékeket (az adott sorrendben):

(1) $\Delta'_j,\ \Delta''_j,\ j = 1, 2, \ldots, n,$
(2) $Q(x)$ cél-függvény értékét az új x pontban, majd
(3) $\Delta_j(x)$ determinánsokat:

$$\Delta_j(x) = \begin{vmatrix} \Delta'_j & Q(x) \\ \Delta''_j & 1 \end{vmatrix},\ j = 1, 2, \ldots, n.$$

A kiszámolt $\Delta_j(x)$ determinánsok elemzése során a következő esetek fordulhatnak elő:

(1) Minden nem-bázis indexű $\Delta_j(x)$ determináns nem-negatív, azaz

$$\Delta_j(x) \geq 0,\ \forall j \in J_N;$$

(2) Létezik legalább egy olyan nem-bázis j_0 index, hogy $\Delta_{j_0}(x)$ negatív, és az összes m darab x_{ij_0}, $i = 1, 2, \ldots, m$; együttható nem-pozitív, azaz

$$J_0 = \{j :\ j \in J_N;\ \Delta_j(x) < 0\} \neq \emptyset;$$

és

$$J_0^- = \{j :\ j \in J_0;\ x_{ij} \leq 0,\ \forall i = 1, 2, \ldots, m\} \neq \emptyset;$$

(3) Létezik legalább egy olyan nem-bázis j_0 index, hogy $\Delta_{j_0}(x)$ negatív, és minden ilyen j_0 számára legalább egy x_{ij_0} együttható pozitív értékű, azaz

$$J_0 = \{j :\ j \in J_N;\ \Delta_j(x) < 0\} \neq \emptyset;$$

és

$$J_0^- = \{j :\ j \in J_0;\ x_{ij} \leq 0,\ \forall i = 1, 2, \ldots, m\} = \emptyset;$$

Az 1. esetben az optimálitási kritériumnak megfelelően (lásd. II.4. Tétel), x vektor optimális megoldása a(z) (30)-(33) hiperbolikus programozási feladatnak. Befejeztük, a feladat meg van oldva.

II.3. Megjegyzés. *Ha $\Delta_j(x)$ determinánsok között van legalább egy nulla értékű, akkor ez azt jelenti, hogy az adott hiperbolikus programozási feladatnak több optimális megoldása van (lásd. (43) képletet).*

A 2. esetben az eredeti (30)-(33) hiperbolikus programozási feladat S lehetséges halmaza nem-korlátos, ezt az esetet pedig a vizsgálatunkból kizártuk (lásd. 31. oldalt). A szimplex módszer nem rendelkezik olyan eszközökkel, amlyek lehetővé tennének korlátlan halmazon optimalizálni tört-alakú célfüggvényt. Tényleg, ebben az esetben létezik legalább egy olyan j_0 index, hogy $\Delta_{j_0}(x) < 0$, és minden m darab x_{ij_0} együttható nem-pozitív, azaz $x_{ij_0} \leq 0,\ i = 1, 2, \ldots, m.$ A(z) (41) képletnek megfelelően ez pedig azt jelenti, hogy θ értéknek nincsen felső korlátja, és emiatt a θ érték tetszőleges nagy lehet. Ahogy ez következik a(z) (39) képletből, ilyenkor az $x(\theta)$ új vektor marad a lehetséges a θ érték nagyságától függetlenül, és ezért tartalmazhat tetszőlegesen nagy $x_j(\theta)$ elemeket. Az utóbbi pedig azt jelenti, hogy az S lehetséges halmaz korlátlan. Itt a szimplex módszert meg kell szakítani, mert a módszer ilyen esetben nem alkalmazható.

II.4. Megjegyzés. *A 2.eset előfordulása nem jelenti azt, hogy az adott hiperbolikus programozási feladat elvileg (definició szerint) nem megoldható. Tényleg, mivel $Q(x)$ cél-függvény tört alakú, ezért lehet, hogy végtelen sok lehetséges x pontban az értéke véges, azaz a*

$$\lim_{\substack{x \to \infty \\ x \in S}} Q(x)$$

értéke lehet véges szám.

Meg kell jegyeznünk, hogy a szimplex módszer a korlátlan lehetséges halmazra való adaptalásával több cikkben kísérleteztek a kutatók az 1970-es években és később (lásd. [**29**], [**96**]), de, sajnos nem nagy sikerrel.

A 3. esetben létezik olyan $x(\theta)$ új lehetéseges bázismegoldás, hogy

$$Q(x(\theta)) \ > \ Q(x).$$

Tényleg, a megtett feltételezéseknek megfelelően az adott esetben megtalálható legalább egy olyan nem-bázis j index, hogy $J_0 \neq \emptyset$ és $J_0^- = \emptyset$. Akkor a(z) (41) tartományból következik, hogy a θ értéke felülről korlátos, és a maximális értéke meghatározható a(z) (42) képletből. Mivel $x(\theta)$ vektor a(z) (30)-(33) hiperbolikus programozási feladat lehetséges megoldása, és $D(x) > 0,\ \forall x \in S,$ ebből következik, hogy $D(x(\theta)) > 0$. Az utóbbi pedig azt jelenti, hogy a nem-üres J_0 halmazból kiválaszthatunk olyan j_0

indexet (lásd. (43) képlet), hogy

$$Q(x(\theta)) - Q(x) = \frac{-\theta\Delta_{j_0}(x)}{D(x(\theta))} > 0.$$

Ebből következik, hogy ha az A_{j_0} vektort vezetjük be a bázisba, akkor olyan $x(\theta)$ új bázismegoldást fogunk kapni, amelynél $Q(x(\theta)) > Q(x)$.

Tehát a 3. esetben az x aktuális lehetséges csúcspontból átugrottunk egy szomszédos $x(\theta)$ lehetséges csúcsbontba, amelyben cél-függvény értéke "jobb" (a maximalizálási feladatnál ez azt jelenti, hogy "nagyobb"). Az új $x(\theta)$ pont előállítását *szimplex-iterációnak* szokták nevezni.

Mivel az S lehetséges megoldások halmaza korlátos, ezért a csúcspontok száma véges. Az utóbbi biztosítja azt, hogy véges számú szimplex-iteráció után kapjuk az 1. vagy 2. esetet.

4. Szimplex tábla

A szimplex módszer végrehajtásakor a megoldandó hiperbolikus programozási feladathoz tartozó adatokat az ún. *szimplex táblázatban* (angolul – *simplex tableau*, oroszul – *"симплексная таблица"*) szokták tartani. Az ilyen szimplex tábla megtekinthető a(z) 1. táblázatban.

			p_0 d_0	p_1 d_1	p_2 d_2	$\cdots$ $\cdots$	p_n d_n
B	P_B	D_B	x_B	A_1	A_2	$\cdots$	A_n
A_{s_1}	p_{s_1}	d_{s_1}	x_{s_1}	x_{11}	x_{12}	$\cdots$	x_{1n}
A_{s_2}	p_{s_2}	d_{s_2}	x_{s_2}	x_{21}	x_{22}	$\cdots$	x_{2n}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
A_{s_m}	p_{s_m}	d_{s_m}	x_{s_m}	x_{m1}	x_{m2}	$\cdots$	x_{mn}
	$P(x)$			Δ'_1	Δ'_2	$\cdots$	Δ'_n
	$D(x)$			Δ''_1	Δ''_2	$\cdots$	Δ''_n
	$Q(x)$			$\Delta_1(x)$	$\Delta_2(x)$	$\cdots$	$\Delta_n(x)$

1. Táblázat. Szimplex táblázat.

A lineáris programozási szimplex táblával szemben a legfontosabb különbségek itt a következők:

(1) Ha lineáris esetben a "B"-vel jelölt oszlop után következik a "P_B" és "x_B" vektor, akkor a hiperbolikus esetben nekünk nyilván kell tartanunk a "D_B" oszlopot is.

(2) Ha lineáris feladat esetén csak p vektorra van szükség, akkor a hiperbolikus esetben a p vektort tartalmazó felső sor alatt még a d vektort kell tartanunk.
(3) Végül lineáris programozási feladatban csak egy Δ-sorra van szükség, viszont a hiperbolikus programozási feladat esetén három Δ-sorra van szükségünk, és ezért a szimplex tábla alsó részén 3 darab Δ-sor található.

5. Az iterációk közötti kapcsolat

Az egyik bázisról a másik bázisra való áttéréssel kapcsolatos műveleteket *szimplex transzfomációnak* (ang. *pivot transformation*) szokták nevezni. Most megvizsgáljuk, hogy az egyik bázisról a másikra való áttérés során hogyan transzfomálódik a szimplex táblázat. Ennek a transzformációnak a lényege látható a(z) 2. táblázatban. A táblázatban az x_{rk}-val jelöltük az

x_{ij}	$\cdots$	x_{ik}		$x_{ij} - \dfrac{x_{rj}\, x_{ik}}{x_{rk}}$	$\cdots$	0
$\vdots$	$\cdots$	$\vdots$	$\longrightarrow$	$\vdots$	$\cdots$	$\vdots$
x_{rj}	$\cdots$	x_{rk}		$\dfrac{x_{rj}}{x_{rk}}$	$\cdots$	1

2. Táblázat. Transzformációs átalakítások a szimplex táblázatban.

ún. *generáló elemet* (ang. *pivot element*), úgy hogy a bázisba bevezetendő változó indexe k és a bázisban levő és cserélendő vektor indexe s_r, azaz az A_{s_r} vektor bázisban való poziciója r.

II.9. Definíció. *A szimplex táblázatban a generáló elem sorát* ***generáló sornak****, a generáló elem oszlopát pedig* ***generáló oszlopnak*** *szokták nevezni (angolul –* ***pivot row*** *és* ***pivot column****, oroszul – "генерирующая строка" és "генерирующий столбец").*

A táblázatban szereplő átalakítások a következők:

(1) A generáló sorhoz tartozó összes elemet osztjuk az x_{rk} ($x_{rk} \neq 0$) generáló elemmel. Ennek következtében az x_{rk} új értéke az 1. Minden más eleme ennek a sornak x_{rj}/x_{rk} értéket kap $j = 1, 2, \ldots, n$.

(2) Minden más sorhoz tartozó összes elem új értéke meghatározható a következő képletből

$$x_{ij} - (x_{rj}\, x_{ik})/x_{rk}.$$

(3) A generáló oszlop minden más eleme (azaz a generáló sorhoz nem tartozó elemekről van szó) új értéke 0, azaz

$$x'_{ik} = x_{ik} - (x_{rk}x_{ik})/x_{rk} = 0.$$

6. A szimplex módszer indítása

Amikor az előző részben tárgyaltuk a szimplex módszer elméleti hátterét, feltételeztük, hogy van egy x induló lehetséges bázismegoldás.

Ahogy már említettük (lásd. 39. oldal), bizonyos esetekben az induló bázismegoldás könnyen előállítható. Pl. ha az A mátrixban van m darab olyan A_j oszlopvektor, amelyekből össze lehet állítani egy $(m \times m)$ méretű egységmátrixot. Világos, hogy a gyakorlatban ilyen feladatok viszonylag ritkán fordulnak elő. Nekünk pedig meg kell tanulnunk, hogyan kell ilyenkor kezelni az általános A mátrixot.

Mielőtt áttérünk az általános esetre, tekintsünk még meg egy "könnyű" esetet.

Tegyük fel, hogy a megoldandó hiperbolikus programozási feladat a fő feltételrendszerében tartalmaz csak " $\leq$ " relációjű feltételeket, azaz

$$\sum_{j=1}^{n} A_j x_j \leq b, \quad x_j \geq 0, \quad j = 1, 2, \ldots, n,$$

és a $b = (b_1, b_2, \ldots, b_m)^T$ vektor minden b_i, $i = 1, 2, \ldots, m$, eleme nemnegatív. Ebben az esetben a kanonikus alakba való átalakításkor a feladatba be kell vezetnünk m darab mesterséges változót:

$$x_{n+1}, \ x_{n+2}, \ \ldots, \ x_{n+m},$$

(lásd. 1.4. szekció, 7. oldal). Ezekhez a mesterséges változókhoz tartozik m darab

$$A_{n+1}, \ A_{n+2}, \ \ldots, \ A_{n+m},$$

egységvektor, amelyek $(m \times m)$ méretű

$$I \ = \ (A_{n+1}, \ A_{n+2}, \ \ldots, \ A_{n+m})$$

egységmátrixot alkotnak, ahol

$$A_{n+i} = (\underbrace{0, 0, \ldots, 0, \overbrace{1}^{i}, 0, 0, \ldots, 0}_{m})^T, \quad i = 1, 2, \ldots, m,$$

azaz

$$A_{n+1} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad A_{n+2} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \ldots, \quad A_{n+m} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$$

Így induló bázismegoldásként használhatjuk a következő vektort:

$$x = (\underbrace{0, 0, \ldots, 0}_{n}, b_1, b_2, \ldots, b_m)^T.$$

II.5. Megjegyzés. *Az ilyen módon bevezetett* $x_{n+1}, x_{n+2}, \ldots, x_{n+m}$ *mesterséges változók a célfüggvényben nem kapnak helyet, azaz összes az* p_j *és* d_j, $j = n+1, n+2, \ldots, n+m$, *nulla értékű. Így kapjuk az átalakított (44)-(46) feladatot.*

$$Q(x) = \frac{\sum_{j=1}^{n} p_j x_j + \sum_{j=n+1}^{n+m} 0x_j + p_0}{\sum_{j=1}^{n} d_j x_j + \sum_{j=n+1}^{n+m} 0x_j + d_0} \longrightarrow \max(\text{ or } \min) \tag{44}$$

a következő feltételek mellett

$$\begin{cases} a_{11}x_1 + & \ldots & +a_{1n}x_n & +x_{n+1} & & & = b_1 \\ a_{21}x_1 + & \ldots & +a_{2n}x_n & & +x_{n+2} & & = b_2 \\ \vdots & & \vdots & & & & \vdots \\ a_{m1}x_1 + & \ldots & +a_{mn}x_n & & & +x_{n+m} & = b_m \end{cases} \tag{45}$$

$$x_j \geq 0, \quad j = 1, 2, \ldots, n+m. \tag{46}$$

Nyilvánvalóvá válik, hogy a(z) (44)-(46) hiperbolikus programozási feladatnak megfelel a(z) 3. táblázatban szereplő induló szimplex tábla.

II.6. Megjegyzés. *Vegyük tudomásul, hogy a(z) 3. táblázatban az* x_{ij} *együtthatók helyett (azaz* x_{ij} *együtthatókként) szerepelnek az eredeti* a_{ij} *koefficiensek. Ez azért van így, mert az induló lehetséges bázisban szereplő* A_{n+i}, $i = 1, 2, \ldots, m$, *vektorok egységvektorok, és emiatt*

$$A_j = \sum_{i=1}^{m} A_{n+i} a_{ij}, \ j = 1, 2, \ldots, n.$$

Sajnos az általános esteben az induló lehetséges bázismegoldás előállítása nem triviális feladat. Éppen ezért a fejezet következő két részében mi két megfelelő módszert fogunk tárgyalni. Ezekből a módszerekből az első az ún. *Nagy M-módszer* (angolul – "*Big M-method*", oroszul – "*M-метод*") és

				p_1	$\cdots$	p_n	0	$\cdots$	0
				d_1	$\cdots$	d_n	0	$\cdots$	0
B	P_B	D_B	x_B	A_1	$\cdots$	A_n	A_{n+1}	$\cdots$	A_{n+m}
A_{n+1}	0	0	b_1	a_{11}	$\cdots$	a_{1n}	1	$\cdots$	0
A_{n+2}	0	0	b_2	a_{21}	$\cdots$	a_{2n}	0	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
A_{n+m}	0	0	b_m	a_{m1}	$\cdots$	a_{mn}	0	$\cdots$	1
	$P(x)$			Δ'_1	$\cdots$	Δ'_n	0	$\cdots$	0
	$D(x)$			Δ''_1	$\cdots$	Δ''_n	0	$\cdots$	0
	$Q(x)$			$\Delta_1(x)$	$\cdots$	$\Delta_n(x)$	0	$\cdots$	0

3. Táblázat. Induló szimplex tábla a(z) (44)-(46) feladathoz.

a második az ún. *Kétfázisú szimplex módszer* (ang.– "*Two-phase simplex method*", oroszul – "*Двух-фазисный симплексный метод*"). Ez a két módszer lehetőséget ad az induló lehetséges bázismegoldás könnyű és problémamentes meghatározására, és mindkettő alkalmazható tetszőleges hiperbolikus programozási feladatnál.
Meg kell jegyeznünk, hogy a lineáris programozásban használt Nagy M-módszer és Kétfázisú szimplex módszer lineáris változatának több speciális implementációja van (lasd. pl.[**32**]). Ezeknek a speciális lineáris programozási változatoknak a többségét könnyen ki lehet terjeszteni a hiperbolikus programozási feladatra is.

6.1. Nagy M-módszer

A módszer alkalmazásakor az eredeti normalizált kanonikus alakú (30)-(33) hiperbolikus programozási feladat fő feltételrendszerében minden i-dik feltételbe be kell vezetni egy-egy mesterséges x_{n+i} $(i = 1, 2, \ldots, m)$ változót, így összesen m darab

$$x_{n+1},\ x_{n+2},\ \ldots,\ x_{n+m}$$

mesterséges változó kerül a feltételrendszerbe. Ezenkívül ezeket a mesterséges változókat be kell vezetni a $P(x)$ függvénybe is $-M$ együtthatóval. Így, az eredeti normalizált kanonikus alakú (30)-(33) hiperbolikus programozási feladat helyett kapjuk a következő *M-feladatot* [**42**]:

$$\tilde{Q}(x) = \frac{P(x) - M\sum_{i=1}^{m} x_{n+i}}{D(x)} \longrightarrow \max \tag{47}$$

$$\sum_{j=1}^{n} a_{ij}x_j + x_{n+i} = b_i, \quad i = 1, 2, \ldots, m, \tag{48}$$

$$x_j \geq 0, \quad j = 1, 2, \ldots, n+m, \tag{49}$$

ahol M egy tetszőlegesen nagy pozitív számot jelöl (ez a szám nem igényel semmiféle konkrét numerikus értéket, csak úgy kell kezelnünk, hogy ez nagyon nagy pozitív szám).

Továbbá az ilyen módon előállított M-feladatnál az induló lehetséges bázismegoldásként használhatjuk a következő

$$x = (\underbrace{0, 0, \ldots, 0}_{n}, b_1, b_2, \ldots, b_m),$$

vektort. Tehát a szimplex módszer indításához szükséges lehetséges bázismegoldás megvan, szimplex módszer ezen az M-feladaton már indítható.

Nyilvánvaló, hogy az induló szimplex táblában

$$\begin{aligned} \Delta'_j &= \sum_{i=1}^{m}(-M)a_{ij} - p_j = -M\sum_{i=1}^{m} a_{ij} - p_j, \\ \Delta''_j &= \sum_{i=1}^{m} 0a_{ij} - d_j = -d_j, \\ \Delta_j(x) &= \Delta'_j - Q(x)\Delta''_j, \end{aligned}$$

ahol $j = 1, 2, \ldots, n$, és $Q(x) = (p_0 - M\sum_{i=1}^{m} x_{n+i})/d_0$.

Tegyük fel, hogy

$$\tilde{x} = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n, \tilde{x}_{n+1}, \ldots, \tilde{x}_{n+m})^T$$

optimális megoldása a(z) (47)-(49) M-feladatnak.

A következő esetek fordulhatnak elő

II.5. Tétel. *Ha $\tilde{x}$ vektor optimális megoldása a(z) (47)-(49) M-feladatnak és*

$$\tilde{x}_{n+i} = 0, \quad i = 1, 2, \ldots, m,$$

azaz

$$\tilde{x} = (\tilde{x_1}, \tilde{x}_2, \ldots, \tilde{x}_n, \underbrace{0, 0, \ldots, 0}_{m}),$$

akkor $x^* = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n)^T$ optimális megoldása az eredeti (30)-(33) feladatnak.

II.6. Tétel. *Ha $\tilde{x}$ vektor optimális megoldása a(z) (47)-(49) M-feladatnak, és a mesterséges változók között van legalább egy pozitív értékű, azaz $\exists i_0 : \tilde{x}_{n+i_0} > 0$, $1 \leq i_0 \leq m$, akkor az eredeti (30)-(33) feladat nem megoldható, mert az S lehetséges halmaza üres, azaz $S = \emptyset$.*

II.7. Megjegyzés. *Az M-feladat S_M lehetséges halmaza nem-üres mert tartalmaz legalább egy*

$$x = (\underbrace{0, 0, \ldots, 0}_{n}, b_1, b_2, \ldots, b_m)$$

vektort, amely kielégíti a(z) (48) és (49) feltételeket. Éppen ezért az az eset, amikor az M-feladat azért nem oldható meg, mert az S_M lehetséges halmaza üres, nem fordulhat elő.

II.8. Megjegyzés. *Az M-feladatban cél-függvényt nem csak a(z) (47) alakban használhatjuk, hanem a következő alakokban is:*

$$\tilde{Q}(x) = \frac{P(x)}{D(x) + M\sum_{i=1}^{m} x_{n+i}} \longrightarrow \max$$

vagy

$$\tilde{Q}(x) = \frac{P(x) - M\sum_{i=1}^{m} x_{n+i}}{D(x) + M\sum_{i=1}^{m} x_{n+i}} \longrightarrow \max .$$

II.9. Megjegyzés. *Az M-feladatban szereplő m darab mesterséges változóra nem mindig van szükség. Nyilvánvaló, hogy ezek a mesterséges változókra csak azért van szukség, hogy a mátrixban legyen m darab egységvektor. Természtesen ha a mátrixban már van néhány egységvektor, akkor csak annyi mesterséges változóra van szükség, hogy az egységvektorok száma pontosan legyen m.*

Tehát ha az eredeti hiperbolikus programozási feladat megoldható, akkor a Nagy M-módszer használatával meghatározhatjuk a feladat optimális megoldását. Ha viszont az eredeti hiperbolikus programozási feladat nem megoldható, mert lehetséges halmaza üres vagy a cél-függvény felülről nem korlátos, akkor a Nagy M-módszer jelezni fogja ezt.

Az M-feladathoz tatozó induló szimplex tábla a 4. táblázatban megtekinthető.

Tekintsük a következő numerikus példát.

Adott a következő általános hiperbolikus programozási feladat.

$$Q(x) \;=\; \frac{P(x)}{D(x)} \;=\; \frac{10x_1 + 5x_2 + 2}{1x_1 + 4x_2 - 4} \longrightarrow \max$$

				p_1	$\cdots$	p_n	$-M$	$\cdots$	$-M$
				d_1	$\cdots$	d_n	0	$\cdots$	0
B	P_B	D_B	x_B	A_1	$\cdots$	A_n	A_{n+1}	$\cdots$	A_{n+m}
A_{n+1}	$-M$	0	b_1	a_{11}	$\cdots$	a_{1n}	1	$\cdots$	0
A_{n+2}	$-M$	0	b_2	a_{21}	$\cdots$	a_{2n}	0	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$
A_{n+m}	$-M$	0	b_m	a_{m1}	$\cdots$	a_{mn}	0	$\cdots$	1
	$P(x)$			Δ_1'	$\cdots$	Δ_n'	0	$\cdots$	0
	$D(x)$			Δ_1''	$\cdots$	Δ_n''	0	$\cdots$	0
	$Q(x)$			$\Delta_1(x)$	$\cdots$	$\Delta_n(x)$	0	$\cdots$	0

4. Táblázat. Nagy M-módszer – Innduló szimplex tábla.

a következő feltételek mellett

$$\begin{aligned} x_1 &- 3x_2 \geq -30 \\ x_1 &+ 2x_2 \geq 20 \\ 2x_1 &- x_2 \leq 10 \end{aligned}$$

$$x_1 \geq 0, \;\; x_2 \geq 0 \; .$$

Határozzuk meg a feladat megoldásához szükséges induló lehetséges bázismegoldást.

Első lépésben alakítsuk át a feladatot standard alakba:

$$\left.\begin{aligned} x_1 &- 3x_2 \geq -30 \\ x_1 &+ 2x_2 \geq 20 \\ 2x_1 &- x_2 \leq 10 \end{aligned}\right\} \quad \Rightarrow \quad \left.\begin{aligned} -x_1 &+ 3x_2 \leq 30 \\ -x_1 &- 2x_2 \leq -20 \\ 2x_1 &- x_2 \leq 10 \end{aligned}\right\}$$

Most konvertáljuk a feltételeket kanonikus alakra. Ehhez vezessünk be szükséges "slack" változókat:

$$\left.\begin{aligned} -x_1 &+3x_2 \leq 30 \\ -x_1 &-2x_2 \leq -20 \\ 2x_1 &-x_2 \leq 10 \end{aligned}\right\} \quad \Rightarrow \quad \left.\begin{aligned} -x_1 &+3x_2 &+x_3 & & & = 30 \\ -x_1 &-2x_2 & &+x_4 & & = -20 \\ 2x_1 &-x_2 & & &+x_5 & = 10 \end{aligned}\right\}$$

Végül normálizáljuk a feltételeket, ehhez szorozzuk meg a második feltétel mindkét oldalát (-1)-gyel:

$$\left.\begin{array}{rrrrrrr} -x_1 & +3x_2 & +x_3 & & & = & 30 \\ x_1 & +2x_2 & & -x_4 & & = & 20 \\ 2x_1 & -x_2 & & & +x_5 & = & 10 \end{array}\right\}$$

Így a következő normál alakú kanononikus feladatot kapjuk:

$$Q(x) = \frac{P(x)}{D(x)} = \frac{10x_1 + 5x_2 + 2}{1x_1 + 4x_2 - 4} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{array}{rrrrrrr} -x_1 & +3x_2 & +x_3 & & & = & 30 \\ x_1 & +2x_2 & & -x_4 & & = & 20 \\ 2x_1 & -x_2 & & & +x_5 & = & 10 \end{array}$$

$$x_1 \geq 0, \;\; x_2 \geq 0 \;\; x_3 \geq 0, \;\; x_4 \geq 0 \;\; x_5 \geq 0$$

A kapott feladatot helyettesítsük a következő M-feladattal:

$$\tilde{Q}(x) = \frac{\tilde{P}(x)}{D(x)} = \frac{10x_1 + 5x_2 - Mx_6 + 2}{1x_1 + 4x_2 - 4} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{array}{rrrrrrrr} -x_1 & +3x_2 & +x_3 & & & & = & 30 \\ x_1 & +2x_2 & & -x_4 & & +x_6 & = & 20 \\ 2x_1 & -x_2 & & & +x_5 & & = & 10 \end{array}$$

$$x_1 \geq 0, \;\; x_2 \geq 0, \;\; x_3 \geq 0, \;\; x_4 \geq 0, \;\; x_5 \geq 0, \;\; x_6 \geq 0.$$

Az ilyen módon kapott M-feladatnál induló lehetséges bázismegoldásként használhatjuk az $x = (0, 0, 0, 30, 0, 10, 20)$ vektort, amelyiknek megfelel a következő bázis: $B = (A_3, A_6, A_5)$.

6.2. Kétfázisú szimplex módszer

A kétfázisú szimplex módszernek csak az első fázisa jelenti az induló lehetséges bázis keresését, maga a módszer már az egész hiperbolikus programozási feladat megoldására szolgál.

A módszer első fázisában a megoldandó normál alakú kanonikus feladat fő feltételrendszerében be kell vezetnünk az $x_{n+1}, x_{n+2}, \ldots, x_{n+m}$ mesterséges változókat. Ugyanúgy, mint az előző részben leírt Nagy M-módszer esetén, most is az a célunk, hogy a mátrixban legyen m darab egységvektor. Ha

be vannak bezetve a mesterséges változók, akkor a következő lineáris programozási feladatot kell megoldanunk:

$$Z(x) = \sum_{i=1}^{m} x_{n+i} \longrightarrow \min \tag{50}$$

a következő feltételek mellett

$$\sum_{j=1}^{n} a_{ij}x_j + x_{n+i} = b_i, \quad i = 1, 2, \ldots, m, \tag{51}$$

$$x_j \geq 0, \quad j = 1, 2, \ldots, n+m. \tag{52}$$

Tekintsük ezt a(z) (50)-(52) *Első fázisú feladatot* (ang. – "*Phase I problem*"). Mivel az

$$x = (\underbrace{0, 0, \ldots, 0}_{n}, b_1, b_2, \ldots, b_m)^T$$

vektor kielégíti a(z) (51)-(52) feltételeket, és (50) cél-függvény alulról korlátos, ebből következik, hogy a(z) (50)-(52) feladat megoldható.

Tegyük fel, hogy az $x' = (x'_1, x'_2, \ldots, x'_n, x'_{n+1}, \ldots, x'_{n+m})^T$ vektor a(z) (50)-(52) optimális megoldása. Ilyenkor a következő két lehetséges eset fordulhat elő:

(1) $Z(x') = 0$, azaz $x'_{n+i} = 0$, $\forall\, i = 1, 2, \ldots, m$. Ebben az esetben
$$x'' = (x'_1, x'_2, \ldots, x'_n)^T$$
vektor az eredeti hiperbolikus programozási feladat lehetséges bázismegoldása. A módszer második fázisában x'' vektor használatával indítjuk a szimplex módszert.

(2) $Z(x') > 0$, azaz $\exists\, i_0 : x'_{n+i_0} > 0$, $1 \leq i_0 \leq m$. Ilyenkor az eredeti hiperbolikus programozási feladat nem megoldható azért, mert lehetséges halmaza üres, azaz $S = \emptyset$.

Tekintsük a következő numerikus példát.

Adott a következő általános hiperbolikus programozási feladat.

$$Q(x) \;=\; \frac{P(x)}{D(x)} \;=\; \frac{5x_1 + 1x_2 + 10}{4x_1 + 2x_2 + 15} \longrightarrow \max$$

a következő feltételek mellett

$$\begin{aligned} x_1 \qquad\quad &\leq \;\; 30 \\ x_1 \qquad\quad &\geq \;\;\; 5 \\ 2x_1 \;+\; x_2 \;&= \;\; 10 \\ x_1 \geq 0, \;\; x_2 &\geq 0\,. \end{aligned}$$

Határozzuk meg a feladat megoldásához szükséges induló lehetséges bázismegoldást.

Alakítsuk át a feladatot kanonikus alakra, ehhez vezessünk be megfelelő nem-negatív mesterséges változókat:

$$\left.\begin{array}{rcl} x_1 & \leq & 30 \\ x_1 & \geq & 5 \\ 2x_1 + x_2 & = & 10 \end{array}\right\} \quad \Rightarrow \quad \left.\begin{array}{rcl} x_1 + x_3 & = & 30 \\ x_1 - x_4 & = & 5 \\ 2x_1 + x_2 & = & 10 \end{array}\right\}$$

A kapott feltételrendszerben már van két darab egységvektor:

$$A_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad \text{és} \quad A_3 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

A hiányzó $(0,1,0)^T$ egységvektor pótlására be kell vezetni még egy mesterséges változót, így a feltételrendszer végleges formája a következő:

$$\left.\begin{array}{rcl} x_1 + x_3 & = & 30 \\ x_1 - x_4 + x_5 & = & 5 \\ 2x_1 + x_2 & = & 10 \end{array}\right\}$$

Végül állítsuk elő az első fázisú lineáris programozási feladatot:

$$Z(x) = x_5 \longrightarrow \min$$

a következő feltételek mellett

$$\begin{array}{rcl} x_1 + x_3 & = & 30 \\ x_1 - x_4 + x_5 & = & 5 \\ 2x_1 + x_2 & = & 10 \end{array}$$

$$x_1 \geq 0, \;\; x_2 \geq 0 \;\; x_3 \geq 0, \;\; x_4 \geq 0 \;\; x_5 \geq 0$$

A kapott feladatnál az induló lehetséges bázismegoldásként használhatjuk az $x = (0, 10, 30, 0, 5)$ vektort, amelyiknek megfelel a következő bázis: $B = (A_3, A_5, A_2)$.

7. A szimplex tábla kompakt formája

A szimplex módszer elméleti hátterének vizsgálata során megtekintett szimplex tábla (lásd. 1. táblázat) a belső részében tartalmazza a feladathoz tartozó összes adatot. Amint kiderült az előző két alfejezetben, a bázishoz tartozó vektoroknak megfelelő oszlopokban a szimplex tábla tartalmazza az $(m \times m)$ méretű egység-matrixot. Nyilvánvaló, hogy ez az egységmátrix az egyik iterációból a másikba való másolása felesleges művelet, ezért a

gyakorlatban és természtesen a számítógépes kódokban használni szokták az ún. *kompakt szimplex táblát*, amely nem tartalmaz egységmátrixot (lásd. 5. táblázat).

		p_1	p_2	$\cdots$	p_n
		d_1	d_2	$\cdots$	d_n
B	x_B	A_1	A_2	$\cdots$	A_n
A_{n+1}	b_1	a_{11}	a_{12}	$\cdots$	a_{1n}
A_{n+2}	b_2	a_{21}	a_{22}	$\cdots$	a_{2n}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
A_{n+m}	b_m	a_{m1}	a_{m2}	$\cdots$	a_{mn}
$P(x)$		Δ'_1	Δ'_2	$\cdots$	Δ'_n
$D(x)$		Δ''_1	Δ''_2	$\cdots$	Δ''_n
$Q(x)$		$\Delta_1(x)$	$\Delta_2(x)$	$\cdots$	$\Delta_n(x)$

5. Táblázat. Kompakt szimplex tábla.

Most megviszgáljuk, hogy az egyik bázisról a másikra való áttérés során hogyan transzformálódik a kompakt szimplex tábla (lásd. 6. táblázat). A transzformáció a következő lépésekből áll:

(1) Az x_{rk} generáló elem helyére annak a reciproka kerül, azaz

$$x_{rk} \;\Rightarrow\; 1/x_{rk}$$

mégpedig $x_{rk} \neq 0$.

(2) A generáló sor többi elemét osztjuk az x_{rk} generáló elemmel:

$$x_{rj} \;\Rightarrow\; x_{rj}/x_{rk}, \quad j = 1, 2, \ldots, n, \quad j \neq k.$$

(3) A generáló oszlop többi elemét osztjuk $(-x_{rk})$ -val, így

$$x_{ik} \;\Rightarrow\; -x_{ik}/x_{rk}, \quad \forall i = 1, 2, \ldots, m, \;\; i \neq r\ .$$

(4) Végül a generáló sorhoz és generáló oszlophoz nem tartozó elemek számára

$$x_{ij} - (x_{rj}\, x_{ik})/x_{rk}.$$

x_{ij}	$\cdots$	x_{ik}		$x_{ij} - \dfrac{x_{rj}\, x_{ik}}{x_{rk}}$	$\cdots$	$-\dfrac{x_{ik}}{x_{rk}}$
$\vdots$	$\cdots$	$\vdots$	$\longrightarrow$	$\vdots$	$\cdots$	$\vdots$
x_{rj}	$\cdots$	x_{rk}		$\dfrac{x_{rj}}{x_{rk}}$	$\cdots$	$\dfrac{1}{x_{rk}}$

6. Táblázat. Szimplex transzformáció kompakt szimplex táblában

Szokásos jelölések használatával írjuk le a transzformációs képleteket:

$$x'_{ij} = \begin{cases} \dfrac{1}{x_{rk}}, & j = k,\ i = r; \\ -\dfrac{x_{ij}}{x_{rk}}, & j = k,\ i = 1, 2, \ldots, m,\ i \neq r; \\ \dfrac{x_{ij}}{x_{rk}}, & j = 1, 2, \ldots, n,\ j \neq k,\ i = r; \\ x_{ij} - \dfrac{x_{rj}\, x_{ik}}{x_{rk}}, & i = 1, 2, \ldots, m,\ i \neq r, \\ & j = 1, 2, \ldots, n,\ j \neq k. \end{cases} \tag{53}$$

Vegyünk észre, hogy a "széles" szimplex tábla oszlopaiban mindig ugyanazok az A_j oszlopvektorok szerepelnek változatlan sorrendben. Ezzel szemben a kompakt szimplex tábla oszlopaiban csak nem-bázis vektorok szereplenek. Így az iterációk során a kompakt szimplex tábla soraiban és oszlopaiban szereplő bázis és nem-bázis vektorok cserélődnek egymással.

Tegyük fel, hogy a megoldandó hiperbolikus programozási feladatban a

$$B = (A_{n+1}, A_{n+2}, \ldots, A_{n+m})$$

az aktuális bázis, és a benne szereplő A_{n+r} vektort kell felcserélnünk az A_k nem-bázis vektorral. Ilyenkor az A_k vektort át kell helyeznünk a k-dik oszlopból az r-dik sorba, közben az A_{n+r} bázis-vektor az r-dik sorból áthelyeződik k-dik oszlopba. Ezt az oszlopcsere folyamatot mutatja a(z) 7. táblázat (a csere előtti állapot) és a(z) 8. táblázat (a csere utáni állapot).

		p_1	$\cdots$	p_k	$\cdots$	p_n
		d_1	$\cdots$	d_k	$\cdots$	d_n
B	x_B	A_1	$\cdots$	A_k	$\cdots$	A_n
A_{n+1}	b_1	x_{11}	$\cdots$	x_{1k}	$\cdots$	x_{1n}
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$
A_{n+r}	b_r	x_{r1}	$\cdots$	x_{rk}	$\cdots$	x_{rn}
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$
A_{n+m}	b_m	x_{m1}	$\cdots$	x_{mk}	$\cdots$	x_{mn}
$P(x)$		Δ'_1	$\cdots$	Δ'_k	$\cdots$	Δ'_n
$D(x)$		Δ''_1	$\cdots$	Δ''_k	$\cdots$	Δ''_n
$Q(x)$		$\Delta_1(x)$	$\cdots$	$\Delta_k(x)$	$\cdots$	$\Delta_n(x)$
				$\Uparrow$		

$\Rightarrow$

7. Táblázat. Kompakt szimplex tábla – Csere előtt.

		p_1	$\cdots$	p_{n+r}	$\cdots$	p_n
		d_1	$\cdots$	d_{n+r}	$\cdots$	d_n
B	x_B	A_1	$\cdots$	A_{n+r}	$\cdots$	A_n
A_{n+1}	b'_1	x'_{11}	$\cdots$	x'_{1k}	$\cdots$	x'_{1n}
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$
A_k	b'_k	x'_{r1}	$\cdots$	x'_{rk}	$\cdots$	x'_{rn}
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$
A_{n+m}	b'_m	x'_{m1}	$\cdots$	x'_{mk}	$\cdots$	x'_{mn}
$P(x)$		Δ'_1	$\cdots$	Δ'_{n+r}	$\cdots$	Δ'_n
$D(x)$		Δ''_1	$\cdots$	Δ''_{n+r}	$\cdots$	Δ''_n
$Q(x)$		$\Delta_1(x)$	$\cdots$	$\Delta_{n+r}(x)$	$\cdots$	$\Delta_n(x)$

8. Táblázat. Kompakt szimplex tábla – Csere után.

8. Be- és kivezetendő változó kiválasztása

A $\Delta_j(x)$ determinánsok vizsgálata során előfordulhat, hogy ezek között a determinánsok között néhány determináns negatív értékű. Ilyenkor felmerül a kérdés - melyik nem-bázis vektort kell bevezetni a bázisba? Továbbá ha sikerült kiválasztani egy ilyen vektort, akkor majd a θ-teszt során előfordulhat, hogy a bázisban több olyen pozició is van, ahova be lehet vezetni a kiválasztott nem-bázis vektort. Ilyenkor felmerül még egy kérdés - melyik bázisvektort kell kivezetni a bázisból? Ezekre a kérdésekre azért kell választ adnunk, mert ettól függ egyrészt a módszer hatékonysága, másrészt az eljárás számítógépes implementálása.

Ebben az alfejezetben éppen az ilyen kérdésekre válaszokat adó szabályokkal foglalkozunk.

8.1. Bevezető szabályok

8.1.1. *Legmeredekebb növekedés szabálya.* Az adott szabály szerint az új bázisba olyan nem-bázis változó kerül, amelyhez a legnagyobb abszolút értékű negatív $\Delta_j(x)$ determináns tartozik. Más szavakkal, ha J_N^- jelöli az olyan j indexek halmazát, amelyeknél $\Delta_j(x)$ determinánsok negatív értékűek, azaz

$$J_N^- = \{j : \ j \in J_N, \ \Delta_j(x) < 0\} \ ,$$

és

$$\Delta_{j_0}(x) = \max_{j \in J_N^-} |\Delta_j(x)|, \tag{54}$$

akkor bázisba kerül az x_{j_0} változó. Az angol nyelvű szakirodalomban ezt a szabályt a "*Steepest Ascent Rule*"-nak nevezik.

8.1.2. *Legnagyobb növekedés szabálya.* Az előző szabály leggyakrabban használt alternatívája az ún. "*Legnagyobb növekedés szabálya*" (ang. – "*highest step method*").

Az adott szabály szerint az a változó kerül a bázisba, amelyiknek a bázisba való bevezetése vezet a cél-függvény értékének legnagyobb növekedéséshez. A(z) (43) képletnek megfelelően (lásd.38). oldal) ez azt jelenti, hogy minden iterációs lépésnél meg kell határozni azt a j_0 indexet, amelynél

$$\max_{j \in J_N^-} \left\{ \frac{-\theta \Delta_j(x)}{D(x(\theta))} \right\} = \frac{-\theta \Delta_{j_0}(x)}{D(x(\theta))} \ ,$$

ahol $D(x(\theta))$-val jelöltük a $D(x)$ nevező új értékét, és majd x_{j_0} változót fogjuk bevezetni a bázisba.

II.10. Megjegyzés. *Az adott szabály átlagosan több időt igényel iterációs lépesenként (a bázisba való bevezetendő változó meghatározása több munkával jár), de gyakran eredményez kisebb számú iterációs lepéseket.*

8.1.3. *Lexikografikus szabályok.* Ezek a szabályok feltételezik, hogy a szimplex módszer indításához a feladatban szereplő változók valamely kritérium szerint rendezve vannak. Ez a rendezési szabály tetszőleges lehet, de ha egyszer kiválasztottunk egy rendezési szabályt, akkor ez fix marad a szimplex módszer befejezéséig. Az adott szabályok szerint a bázisba bevezethető változók közül az a változó kerül a bázisba, amelyik a rendezési sorrend szerint a *legbaloldalibb* ("*leftmost of the eligible variables rule*"), vagy *legjobboldalibb* ("*rightmost of the eligible variables rule*"). Pédául ha rendezési

szabályként választottuk a j indexet, növekvőleg, akkor az az x_{j_0} változó kerül a bázisba, amely számára teljesül a

$$j_0 = \min_{j \in J_N^-} j, \qquad \text{ahol } J_N^- = \{j : \ j \in J_N, \ \Delta_j(x) < 0\}$$

feltétel.

II.11. Megjegyzés. *Ha a megoldandó hiperbolikus programozási feladat több optimális megoldással rendelkezik, a fenti szabályok általában más-más optimális megoldáshoz vezetnek.*

A fenti szabályok hatékonyságának összehasonlítására 1963-ban két amerikai matematikus H.W. Kunh és R.E.Quandt végeztek számítógépes kísérleteket. Ezeknek a kísérleteknek a során véletlenszám-generátorral előállított lineáris programozási feladatok sorozatán hajtották végre mind a három eljárást, és összehasonlították az iterációs lépések számát, valamint a futási időket. A kapott eredmények azt mutatták, hogy a legmeredekebb növekedés szabály alkalmazása átlagosan alacsonyabb iterációszámhoz vezet és ebben az esetben a futási idők átlaga is kisebb.

8.2. Kivezető szabályok

A szimplex módszer végrehajtása során előfordulhat, hogy a bázisba való bevezetendő változó meghatározása után az ún. θ-teszt (lásd. (42) képlet) nem eredményez egyetlen egy olyan vátozó indexét sem, amelyet ki kellene vezetni a bázisból. A következő szabályokat alkalmazhatjuk ilyenkor.

8.2.1. *Legfelsőbb sor szabály.* Ennek a szabálynak megfelelően azt a sor választjuk a szimplex táblában, amely a legfelső poziciót foglalja el.

8.2.2. *Lexikografikus szabályok.* A lexikografikus szabályok használat esetén a szimplex módszer indítása előtt ki kell választanunk egy rendezési szabályt, amelynek megfelelőn választjuk ki majd a bázist elhagyó változókat.

Hiperbolikus programozási feladatokban alkalmazhatjuk a lineáris programozási feladatokra kifejlesztett szabályokat is. Azokról pedig részletes leírásokat találhatunk a következő forrásokban: [**100**], [**123**], [**129**], [**146**], [**183**], [**191**].

9. Degeneráció és ciklizálás

A jelen jegyezet minden előző részében feltételeztük, hogy az aktuális lehetséges bázismegoldás nem-degenerált (lásd. II.6. Definíció). Előfordulhat azonban, hogy az aktuális lehetséges bázismegoldás tartalmaz legalább egy nulla-értékű bázisváltozót, azaz a megoldás degenerált. Ez a jelenség pedig ahhoz vezethet, hogy a szimplex módszer végrehajtása során többször fordul elő ugyanaz a bázis. Ilyenkor azt mondjuk, hogy a módszer *ciklizál* (ang. –"*cycling*").

Szerencsére a szimplex módszert úgy lehet finomítani, hogy a ciklizálás ne forduljon soha elő [**33**], [**54**].

A gyakorlatban a degenerálás nem szükségképpen vezet mindig ciklizáláshoz, és maga a ciklizálás előforulása rendkívüli ritka esemény [**120**], [**121**].

A szakirodalomban található "ciklizáló" lineáris programozási feladatok többsége "mesterséges" jellegű, azaz a kutatók által tudatosan konstruált feladatok [**157**]. Elképzelhető, hogy a hiperbolikus programozásban is lehet konstruálni "ciklizáló" feladatokat.

Azonban léteznek speciális feladatok, ahol degenerálás és ciklizálás viszonylag gyakori esemény, pl. a sorbanállási elmélet speciális feladatai [**121**].

A szimplex módszer elméleti hátterének targyalásánál feltételeztük, hogy

(1) a kiválasztott j indexre a(z) (42) θ-teszt pozitív értéket eredményez, azaz
$$\theta \;=\; \min_{x_{ij}>0} \frac{x_{s_i}}{x_{ij}} \;=\; \frac{x_k}{x_{kj}} \;>\; 0.$$
(2) A bázisból kivezetendő változó (vagy másképpen a generáló sor) indexét egyedi módon sikerült azonosítani, azaz
$$\frac{x_k}{x_{kj}} < \frac{x_{s_i}}{x_{ij}}, \;\; \forall i = 1, 2, \ldots, m, \;\; s_i \neq k.$$
(3) Minden iterációs lépesben a cél-függvény értéke szigorúan növekszik.

Ha az aktuális lehetséges bázismegoldás nem-degenerált, akkor az 1. számú feltételezés azt jelenti, hogy a bázisba kerülő x_j változó értékül szigorúan pozitív θ-t kap, és ennek köszönhetően az új bázismegoldás szintén nemdegenerált lesz. Ebben az esetben, ha az x_j változóhoz tartozó $\Delta_j(x)$ determináns szigorúan negatív értékű, akkor a(z) (43) képletnek megfelelően a $Q(x)$ cél-függvény értéke szigorúan növekszik, és ezért teljesül a 3. számú feltételezésünk.

Ha így alakul az összes iterációs lépes, nyilvánvaló, hogy a véges számú iterációk után az optimális megodást kapjuk. Szóval, ilyenkor a ciklizálás nem fordulhat elő.

Például tegyük fel, hogy a megoldandó hiperbolikus programozási feladatban szerepel 10 darab változó, és 5 főfeltétel. Továbbá tegyük fel, hogy a feladat összes lehetséges bázismegoldása nem-degenerált. Az ilyen feladat legfeljebb

$$C_{10}^5 = \frac{10!}{5!\,(10-5)!} = 252$$

darab bázismegoldással rendelkezik. Mivel az összes lehetséges bázismegoldás nem-degenerált, és emiatt a szimplex módszer nem ciklizál, ezért legfeljebb 252 iterációs lépés végrehajtása után kapjuk az optimális megoldást.

A 2. számú feltételezésünk biztosítja a degenerálás elkerülését. Tényleg, tegyük fel, hogy

$$\theta = \min_{x_{ij}>0} \frac{x_{s_i}}{x_{ij}} = \frac{x_{s_1}}{x_{1j}} = \frac{x_{s_2}}{x_{2j}}.$$

Ebből következik, hogy az x_j új bázisváltozó az új bázisban vagy az x_{s_1} változó helyét vagy az x_{s_2}-ét foglalja. Ilyenkor ha x_{s_1} (vagy x_{s_2}) változó kikerül a bázisból, és x_{s_2} (vagy x_{s_1}) vátozó a bázisban marad, akkor az x_{s_2} (vagy x_{s_1}) bázisváltózó új értéke nulla lesz. Szóval az új bázismegoldás degenerálódik.

1977-ben R.G.Bland [**33**] kifejlesztett a lineáris programozási feladatkra egy olyan szabályt, amely használja a rendezett indexű változókat, és ezzel biztosítja a ciklizálás elkerülését. Szerencsénkre ez a szbály alkalmazható a hiperbolikus programozásban is.

Bland szabály (Legkisebb index szabály – "Least index rule")

- Ha több, mint egy nem-bázis változó vezethető be abázisba, akkor válasszuk a legkisebb indexűt.
- Ha több, mint egy változó kerülhet ki a bázisból, akkor válasszuk a legkisebb indexűt.

Ezen kívül, R.G.Bland megfogalmazta és bebizonyította [**33**] a következőt

II.7. Tétel. *A fenti szabály alkalmazása esetén a szimplex módszer nem ciklizálhat, és ezért véges számú itárációs lépés után véget ér.*

III. Fejezet
Dualitás

A matematikai programozás elmélete szerint minden folytonos (azaz csak folytonos változókat tartalmazó) optimalizálási feladat számára konstruálható egy ún. *duális feladat*. A két feladat közötti kapcsolatból eredő tulajdonságokat, állításokat számos elméleti és alkalmazási területen használják fel. A dualitás elve megjelenik a különböző matematikai, fizikai, statisztikai ágakban is. A dualitás alkalmazása különösen hasznos a lineáris és hiperbolikus programozásban, amikor a feladat optimális megoldását kell elemezni.

Ebben a fejezetben a hiperbolikus programozásban ismert legfontosabb dualitási eredményeket fogjuk tanulmányozni.

1. A dualitás rövid áttekintése

Ebben a szekcióban röviden áttekintjük a hiperbolikus programozási duálitás néhány megközelítését. Meg kell jegyeznünk, hogy az 1960-as és 1970-es években a duálitási kérdése a hiperbolikus programozásban nagy érdeklődést keltett a kutatók körében, és különböző megközelítésben számos eredményt hozott (lásd. például: [**1**], [**25**], [**26**], [**27**], [**30**], [**41**], [**45**], [**111**], [**163**] [**164**], [**167**], [**168**] [**170**], [**177**], [**178**]). Ezekből a megközelítésekből nem mindegyik bizonyult hasznosnak és alkalmazhatónak.
Azonban a Charnes–Cooper-transzormáción (lásd. [**40**] és a jelen jegyzetben 3. alfejezet), illetve a Gol'stein-féle Lagrange-függvény használatán alapuló megközelítések nagyon hasznosak lettek.

Tekintsük a következő standard hiperbolikus programozási feladatot:

$$Q(x) = \frac{P(x)}{D(x)} = \frac{\sum_{j=1}^{n} p_j x_j + p_0}{\sum_{j=1}^{n} d_j x_j + d_0} \to \max, \tag{55}$$

a következő feltételek mellett

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i, \quad i = 1, 2, \ldots, m, \tag{56}$$

$$x_j \geq 0, \quad j = 1, 2, \ldots, n. \tag{57}$$

A(z) (55)-(57) feladatot "primál" feladatnak fogjuk nevezni (ang.– "primal problem").

A Charnes–Cooper-transzformáció szerint a(z) (55)-(57) feladatnak megfelel a következő lineáris analóg:

$$L(t) = \sum_{j=0}^{n} p_j t_j \longrightarrow \max \tag{58}$$

a következő feltételek mellett

$$\sum_{j=0}^{n} d_j t_j = 1. \tag{59}$$

$$-b_i t_0 + \sum_{j=1}^{n} a_{ij} t_j \leq 0, \quad i = 1, 2, \ldots, m, \tag{60}$$

$$t_j \geq 0, \quad j = 0, 1, 2, \ldots, n, \tag{61}$$

ahol

$$t_j = \frac{x_j}{D(x)}, \;\; j = 1, 2, \ldots, n, \qquad t_0 = \frac{1}{D(x)}.$$

Meg kell jegyeznünk, hogy a(z) (58)-(61) feladat lineáris programozási feladat. A lineáris programozási dualitás elmélete szerint a(z) (58)-(61) feladathoz a következő duális feladat tartozik:

$$\psi(y) = y_0 \longrightarrow \min \tag{62}$$

a következő feltételek mellett

$$d_0 y_0 - \sum_{i=1}^{m} b_i y_i \geq p_0, \tag{63}$$

$$d_j y_0 + \sum_{i=1}^{m} a_{ij} y_i \geq p_j, \quad j = 1, 2, \ldots, n. \tag{64}$$

$$y_i \geq 0, \;\; i = 1, 2, \ldots, m. \tag{65}$$

A továbbiakban ezt a(z) (62)-(65) lineáris programozási feladatot "*duális*" feladatnak fogjuk nevezni a(z) (55)-(57) hiperbolikus programozási feladat számára.

Az 1970-es években Gol'stein [**78**], [**79**] által bevezett törtalakú

$$L(x,y) = \frac{P(x) + \sum_{i=1}^{m} y_i \, (b_i - \sum_{j=1}^{n} a_{ij} \, x_j)}{D(x)}. \tag{66}$$

Lagrange-függvény szintén a(z) (62)-(65) feladathoz vezet.

Most tekintsük a következő kanonikus alakú hiperbolikus programozási feladatot:

$$Q(x) = \frac{P(x)}{D(x)} = \frac{\sum_{j=1}^{n} p_j x_j + p_0}{\sum_{j=1}^{n} d_j x_j + d_0} \to \max, \tag{67}$$

a következő feltételek mellett

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1, 2, \ldots, m, \tag{68}$$

$$x_j \geq 0, \quad j = 1, 2, \ldots, n. \tag{69}$$

Ennek a feladatnak a kövekező duális feladat felel meg:

$$\psi(y) = y_0 \longrightarrow \min \tag{70}$$

a következő feltételek mellett

$$d_0 y_0 - \sum_{i=1}^{m} b_i y_i \geq p_0, \tag{71}$$

$$d_j y_0 + \sum_{i=1}^{m} a_{ij} y_i \geq p_j, \quad j = 1, 2, \ldots, n. \tag{72}$$

Jegyezzük meg, hogy a(z) (70)-(72) nem tartalmaz nem-negatívitási feltételeket.

Végül fogalmazzuk meg a duális feladatot az általános hiperbolikus programozási feladat számára. Legyen a következő általános hiperbolikus programozási feladat:

$$Q(x) = \frac{P(x)}{D(x)} = \frac{\sum_{j=1}^{n} p_j x_j + p_0}{\sum_{j=1}^{n} d_j x_j + d_0} \longrightarrow \max \tag{73}$$

a következő feltételek mellett

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i, \quad i = 1, 2, \ldots, m_1, \tag{74}$$

$$\sum_{j=1}^{n} a_{ij}x_j = b_i, \quad i = m_1 + 1, m_1 + 2, \ldots, m, \tag{75}$$

$$x_j \geq 0, \quad j = 1, 2, \ldots, n_1, \tag{76}$$

ahol $m_1 \leq m, \quad n_1 \leq n$. Ennek a feladatnak következő duális feladat felel meg:

$$\psi(y) = y_0 \longrightarrow \min \tag{77}$$

a következő feltételek mellett

$$d_0 y_0 - \sum_{i=1}^{m} b_i y_i \geq p_0, \tag{78}$$

$$d_j y_0 + \sum_{i=1}^{m} a_{ij} y_i \geq p_j, \quad j = 1, 2, \ldots, n_1. \tag{79}$$

$$d_j y_0 + \sum_{i=1}^{m} a_{ij} y_i = p_j, \quad j = n_1 + 1, n_1 + 2, \ldots, n. \tag{80}$$

$$y_i \geq 0, \quad i = 1, 2, \ldots, m_1. \tag{81}$$

2. Fő tételek

Ebben a szekcióban megfogalmazzuk a dualitás legfontosabb állításait.

III.1. Lemma (Dualitás 1. lemmája). *Ha az*

$$x = (x_1, x_2, \ldots, x_n)^T$$

vektor lehetséges megoldása a(z) (55)-(57) standard hiperbolikus programozási feladatnak, és az

$$y = (y_0, y_1, y_2, \ldots, y_m)$$

vektor lehetséges megoldása a(z) (62)-(65), akkor teljesül a

$$Q(x) \leq \psi(y)$$

egyenlőtlenség.

III.2. Lemma (Dualitás 2. lemmája). *Ha az*

$$x^* = (x_1^*, x_2^*, \ldots, x_n^*)^T$$

vektor lehetséges megoldása a(z) (55)-(57) standard hiperbolikus programozási feladatnak, és az

$$y^* = (y_0^*, y_1^*, y_2^*, \ldots, y_m^*)$$

vektor lehetséges megoldása a(z) (62)-(65) feladatnak, és teljesül a

$$Q(x^*) = \psi(y^*) \tag{82}$$

egyenlőség, akkor az x^* vektor optimális megoldása a(z) (55)-(57) feladatnak, és az y^* vektor optimális megoldása a(z) (62)-(65) feladatnak.

A következő lemma összeköti a primál és duál feladat megoldhatóságát.

III.3. Lemma (Dualitás 3. lemmája). *Ha (62)-(65) duális feladat $\psi(y)$ cél-függvénye az Y lehetséges halmazon alulról nem korlátos, akkor a(z) (55)-(57) primál feladat nem megoldható, mert S lehetséges halmaza üres, azaz $S = \emptyset$.*
Fordítva,
Ha (55)-(57) primál feladat $Q(x)$ cél-függvénye az S lehetséges halmazon felülről nem korlátos, akkor a(z) (62)-(65) duális feladat nem megoldható, mert Y lehetséges halmaza üres, azaz $Y = \emptyset$

III.1. Tétel (Duálitás 1. Tétele). *Ha a(z) (55)-(57) primál hiperbolikus programozási megoldható, és x^* vektor az optimális megoldása, akkor a(z) (62)-(65) duális feladat is megoldható, és tetszőleges y^* optimális megoldása mellett teljesül a*

$$Q(x^*) = \psi(y^*). \tag{83}$$

egyenlőség.
Fordítva,
Ha a(z) (62)-(65) duális feladat megoldható, és y^* vektor az optimális megoldása, akkor a(z) (55)-(57) primál hiperbolikus programozási is megoldható, és tetszőleges x^* optimális megoldása mellett teljesül a(z) (83) egyenlőség.

Megfogalmazunk néhány állítást, amely következik a III.1. Tételből.

III.1. Következmény. *Ahhoz, hogy a(z) (55)-(57) primál és a(z) (62)-(65) duális feladat megoldható legyen, szükséges és elégséges, hogy mindkét feladatnak legyen legalább egy-egy lehetséges megoldása.*

III.2. Következmény. *Ha az x^* vektor a(z) (55)-(57) primál feladat lehetséges megoldása, és az y^* vektor a(z) (62)-(65) duális feladat lehetséges megoldása, akkor ahhoz, hogy x^* és y^* vektor legyen a saját feladatának optimális megoldása szükséges és elégséges, hogy teljesüljön a*

$$Q(x^*) = \psi(y^*)$$

egyenlet.

III.3. Következmény. *Ha (55)-(57) primál feladat (lineáris analóg (58)-(61)) megoldható, akkor a hozzátartozó (58)-(61) lineáris analógja (hozzátartozó (55)-(57) hiperbolikus programozási feladat) is megoldaható. Ezenkívül ha az x^* vektor a(z) (55)-(57) primál feladat optimális megoldása, és a t^* vektor a(z) (58)-(61) lineáris analóg optimális megoldása, akkor*

$$Q(x^*) = \phi(t^*),$$

és

$$\text{ha} \quad t_0^* > 0, \quad \text{akkor} \quad x_j^* = \frac{t_j^*}{t_0^*}, \quad j = 1, 2, \ldots, n; \tag{84}$$

$$\text{ha} \quad t_0^* = 0, \quad \text{akkor} \quad x_j^* = \begin{cases} t_j^* \lim\limits_{k\to\infty} \lambda_k, & j \in J', \\ 0, & j \in J'' \end{cases} \tag{85}$$

III.1. Definíció. *A(z) (56) és (60), azaz*

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad \text{és} \quad -b_i t_0 + \sum_{j=1}^{n} a_{ij} t_j \le 0, \quad i = 1, 2, \ldots, m,$$

feltételekből az i index szerint összeállított feltételpárokat, illetve a(z) (57), (61), azaz

$$x_j \ge 0 \quad \text{és} \quad t_j \ge 0, \quad j = 1, 2, \ldots, n,$$

feltételekből a j index alapján összeállított feltételpárokat ***analóg feltételpároknak*** *fogjuk nevezni.*

III.2. Definíció. *A(z) (56) és (65), azaz*

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad \text{és} \quad y_i \ge 0, \quad i = 1, 2, \ldots, m,$$

feltételekből az i index szerint összeállított feltételpárokat, illetve a(z) (57), (64), azaz

$$x_j \ge 0 \quad \text{és} \quad d_j y_0 + \sum_{i=1}^{m} a_{ij} y_i \ge p_j, \quad j = 1, 2, \ldots, n,$$

feltételekből a j index alapján összeállított feltételpárokat ***duális feltételpároknak*** *fogjuk nevezni.*

III.3. Definíció. *Azt fogjuk mondani, hogy egy (56) (vagy (57)) feltétel* ***rögzített****, ha tetszőleges x^* optimális megoldás mellet az adott feltétel teljesül mint szigorú egyenlőség.*

III.4. Definíció. *Azt fogjuk mondani, hogy egy (56) (vagy (57)) feltétel* ***szabad****, ha legalább egy x^* optimális megoldás mellet az adott feltétel teljesül mint szigorú egyenlőtlenség.*

Hasonló módon ezt a két definíciót kiterjeszthetjük a duális feladat (63), (64), (65), azaz

$$d_0 y_0 - \sum_{i=1}^{m} b_i y_i \geq p_0,$$

$$d_j y_0 + \sum_{i=1}^{m} a_{ij} y_i \geq p_j, \quad j = 1, 2, \ldots, n,$$

$$y_i \geq 0, \quad i = 1, 2, \ldots, m,$$

feltételeire, illetve a lineáris analóg (60), (61), azaz

$$-b_i t_0 + \sum_{j=1}^{n} a_{ij} t_j \leq 0, \quad i = 1, 2, \ldots, m,$$

$$t_j \geq 0, \quad j = 0, 1, 2, \ldots, n,$$

feltételeire is.

III.2. Tétel. *Ha a(z) (55)-(57) primál hiperbolikus programozási feladat és a hozzátartozó (58)-(61) lineáris analóg megoldható, akkor minden analóg feltételpárban mindkét feltétel vagy rögzített vagy szabad.*

III.3. Tétel. *Ha a(z) (55)-(57) primál hiperbolikus programozási feladat és a hozzátartozó (62)-(65) duális feladat megoldható, akkor minden duális feltételpárban az egyik feltétel rögzített, a másik pedig szabad; vagy más szavakkal*

$$\left(\sum_{j=1}^{n} a_{ij} x_j - b_i\right) y_i \;=\; 0, \quad i = 1, 2, \ldots, m,$$

és

$$\left(p_j - d_j y_0 + \sum_{i=1}^{m} a_{ij} y_i\right) x_j \;=\; 0, \quad j = 1, 2, \ldots, n.$$

Jegyzzük meg, hogy a(z) (62)-(65) duális feladatban $n+1$ darab főfeltétel van. Ebből 1 darab (63) feltétel és n darab (64) feltétel. A(z) (64) feltételek a j index szerint a(z) (57) feltélekkel vannak kapcsolatban, és a(z) (65) és (56) feltételek össze vannak kapcsolva az i indez alapján. Duális feladatban egyetelen egy (63) feltételnek nincsen párja a primál feladatban. Ezt a hiányt pótolja a következő állítás:

III.4. Tétel ([8]). *Ha a(z) (55)-(57) primál feladat és a hozzátartozó (62)-(65) duális feladat megoldható, akkor (63) feltétel akkor és csak akkor rögzített, ha a primál feldatnak van legalább egy véges értékű optimális megoldása, azaz*

$$x_j^* < \infty, \quad j = 1, 2, \ldots, n \tag{86}$$

3. Duális változók és stabilitási analízis

Tekintsük a következő kanonikus alakú hiperbolikus programozási feladatot:

$$Q(x) = \frac{P(x)}{D(x)} = \frac{\sum_{j=1}^{n} p_j x_j + p_0}{\sum_{j=1}^{n} d_j x_j + d_0} \longrightarrow \max, \tag{87}$$

a következő feltételek mellett

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1, 2, \ldots, m, \tag{88}$$

$$x_j \geq 0, \quad j = 1, 2, \ldots, n\ . \tag{89}$$

Feltételezzük, hogy

$$D(x) > 0, \ \ \forall x \in R^+ = \{R^n | x \geq 0\}.$$

Tegyük fel, hogy az $x^* = (x_1^*, x_2^*, \ldots, x_n^*)^T$ vektor a (87)-(89) feladat optimális megoldása, és

$$x_j^* < \infty, \ \ j = 1, 2, \ldots, n. \tag{90}$$

Jelöljük $HP(b)$-vel a(z) (87)-(89) feladatot, és cseréljük ki a $HP(b)$ feladatban a $b = (b_1, b_2, \ldots, b_m)^T$ vektort egy másik $b' = (b'_1, b'_2, \ldots, b'_m)^T$ vektorral. Az ilyen módon összeállított új hiperbolikus programozási feladatot $HP(b')$-vel fogjuk jelölni.

A ilyen módon bevezetett két hiperbolikus programozási feladat közötti kapcsolatot a következő állítás írja le:

III.5. Tétel ([9]). *Tegyük fel, hogy a $HP(b)$ feladat megoldható, és x^* vektor ennek a feladatnak nem-degenerált, véges, optimális meoldása. Ha*

$$|b_i - b'_i| < \varepsilon, \quad i = 1, 2, \ldots, m,$$

akkor a $HP(b')$ feladat is megoldható, és tetszőleges x' optimális megoldása mellett teljesül a következő egyenlőség

$$Q(x') = y_0^* + \frac{\sum_{i=1}^{m} y_i^* \ (b'_i - b_i)}{D(x')}, \tag{91}$$

ahol ε elegendően kicsi szám, és az

$$y^* \ = \ (y_0^*, y_1^*, y_2^*, \ldots, y_m^*)$$

vektor a $HP(b)$ feladathoz tartozó duális feladat optimális megoldása.

Jegyezzük meg, hogy a(z) (90) feltételezésnek nagyon fontos és meghatározó szerepe van, mivel G.R.Bitran és T.L.Magnanti [**30**], [**31**] megmutatták, hogy abban az esetben, ha (90) feltétel nem teljesül, akkor a jobb oldali vektorban történő valami kis mértékű változás nem változtatja a cél-függvény optimális értékét.

IV. Fejezet
Érzékenység vizsgálata

A gyakorlati alkalmazások szempontjából gyakran szükséges tudni, hogy a feladatban szereplő együtthatók változásának milyen hatása van a cél-függvény optimális értékére, vagy meddig marad ilyenkor az optimális megoldás optimális.

Ebben a fejezetben az ún. *érzékenység vizsgálatával* (ang.–*"stability analysis"* vagy *"sensitivity analysis"*) fogunk foglalkozni, azaz hogyan hat az együtthatók változtatása az optimális megoldásra.

Először a 1. szekcióban egy numerikus példa segítségével illusztráljuk a jobb oldali b vektorban történő változások hatását az optimális megoldásra, majd az ez után következő 2.-6. részekben megtekintjük a hiperbolikus programozási feladat különböző részeiben történő változási eseteket.

Legyen a következő kanonikus hiperbolikus programozási feladat:

$$Q(x) = \frac{P(x)}{D(x)} = \frac{\sum_{j=1}^{n} p_j x_j + p_0}{\sum_{j=1}^{n} d_j x_j + d_0} \longrightarrow \max \tag{92}$$

a következő feltételek mellett

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1, 2, \ldots, m, \tag{93}$$

$$x_j \geq 0, \quad j = 1, 2, \ldots, n, \tag{94}$$

ahol $D(x) > 0$, $\forall x \in S$.

Tegyük fel, hogy a(z) (92)-(94) feladat megoldható, és az x^* vektor ennek a feladatnak optimális bázismegoldása. Az általánosság megszorítása nélkül feltételezhetjük, hogy

$$x^* = (x_1^*, x_2^*, \ldots, x_m^*, 0, 0, \ldots, 0)^T$$

és ennek a vektornak megfelel a következő bázis:

$$B \ = \ (A_1, A_2, \ldots, A_m),$$

ahol

$$A_j \;=\; (a_{1j}, a_{2j}, \ldots, a_{mj})^T, \quad j = 1, 2, \ldots, n.$$

A fejezet további részeiben mindenütt használni fogjuk a következő jelöléseket:

összes indexhalmaz:	$J = \{1, 2, \ldots, n\}$,
bázis indexek halmaza:	$J_B = \{1, 2, \ldots, m\}$,
nem-bázis indexek halmaza:	$J_N = J \setminus J_B = \{m+1, m+2, \ldots, n\}$.

1. Grafikus bevezetés

Tekintsük a következő numerikus példát:

$$Q(x) = \frac{6\,x_1 + 3\,x_2 + 6}{5\,x_1 + 2\,x_2 + 5} \longrightarrow \max(\min) \tag{95}$$

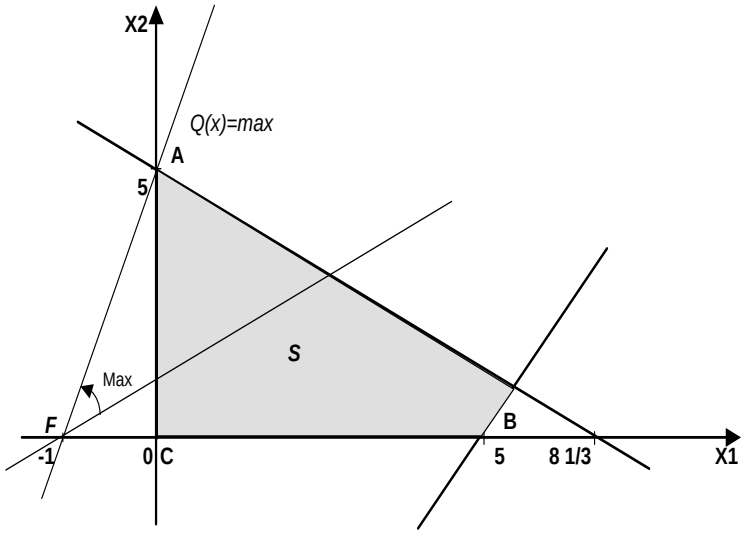
a következő feltételek mellett

$$\begin{cases} 4\,x_1 \;-\; 2\,x_2 \;\leq\; 20\,, & (i) \\ 3\,x_1 \;+\; 5\,x_2 \;\leq\; 25\,, & (ii) \\ \quad x_1 \geq 0, \;\; x_2 \geq 0. \end{cases} \tag{96}$$

Ennek a feladatnak optimális megoldása az

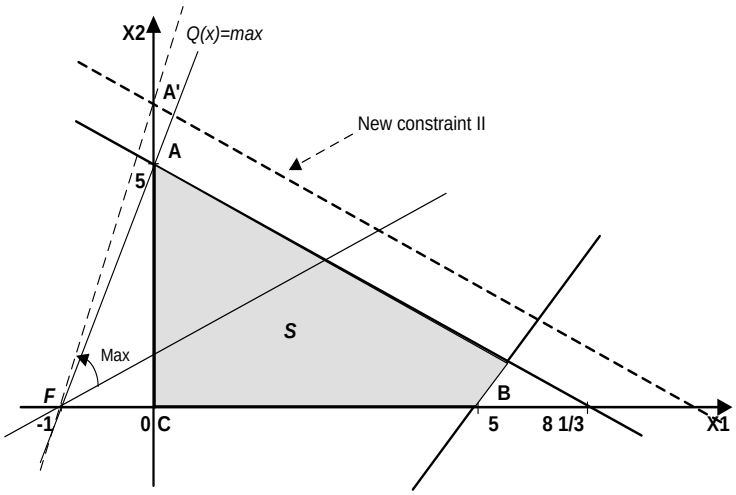
$$x^* = (0, 5), \;\; Q(x^*) = \frac{21}{15},$$

vektor (az A betűvel jelölt pont a(z) 1. ábrán).



1. ábra. (95)-(96) eredeti feladat lehetséges halmaza.

Változtassuk meg jobb oldali $b = (20, 25)^T$ vektor úgy, hogy $b_1 = 20$ maradjon változtatlan, és a $b_2 = 25$ helyett pedig legyen $b_2' = b_2 + \delta = 25 + 5 = 30$. Nyilvánvaló, hogy ennek a valtozásnak nincsen semmi hatása az F fokusz pontra, és az nem változik. Viszont ilyenkor változik a lehetséges halmaz (lásd. 2. ábra). A(z) 2. ábrából látszik, hogy a jobb oldali b vektorba



2. ábra. Megváloztatott lehetséges halmaz.

bevezetett változások miatt a lehetséges halmaz megváltozott úgy, hogy az optimális bázis marad ugyanaz, de az optimális megoldás elmozdult, és áthelyeződött az A' pontba, így az új optimális megoldás az $x' = (0, 6)^T$ vektor lett, és $Q(x') = 24/17$.

Vegyük észre, hogy ebben a numerikus példában a b vektor b_2 elemébe bevezetett δ értéknek nincsen hatása az optimális bázisra, és ezért a δ érték (és emiatt a b_2 érték) lehet tetszőleges nagy. Tehát a δ érték felső korlátja végtelen nagy, azaz ∞. Azonban ha b_2 csökken (azaz $\delta < 0$), akkor az optimális bázis csak addig marad stabil, amig $b_2 + \delta \geq 0$, mivel a negatív értékű b_2 mellett a feladat mogoldhatatlanná válik az üres lehetséges halmaz miatt.

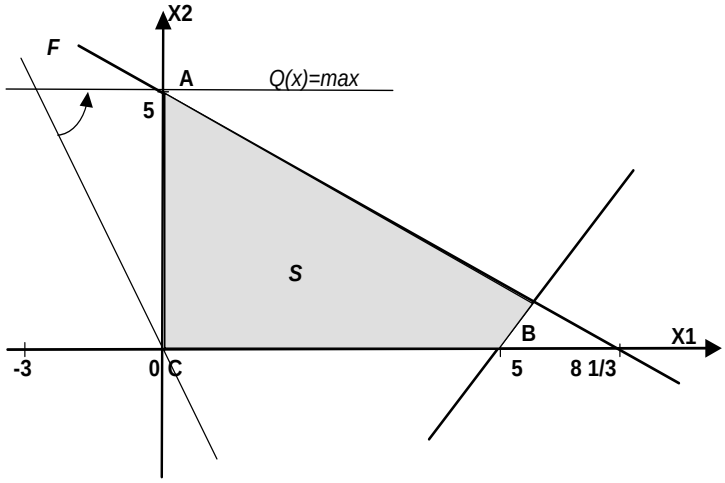
Az utóbbi azt jelenti, hogy $\delta \geq 0 - b_2 = -25$. Végül kapjuk a következő stabilitási tartományt:

$$-25 \leq \delta < \infty \quad \text{vagy} \quad 0 \leq b_2 < \infty \, .$$

Hasonló módon kideríthetjük, hogy a b_1 elem számára a stabilitási tartomány a következő:

$$-30 \leq \delta < \infty \quad \text{vagy} \quad -10 \leq b_1 < \infty \, .$$

Ha változások történnek a cél-függvényben, akkor a helyzet jóval komplikáltabbá válik, mivel az ilyen változások hatnak az F fokus pontra (lásd. 3. ábra), és emiatt olyan lényeges változásokat eredményezhetnek a feladatban, amelyek részletes vizsgálatokat igényelhetnek. Ezenkívül a cél-függvény nevezőjében bevezetett változások miatt előfordulhat a $D(x) > 0\ \forall x \in S$ feltétel megsértése is. Éppen ezért a hiperbolikus programozási feladat különböző részeiben történő változásokat külön-külön alfejezetekben fogjuk tárgyalni.



3. ábra. Megváltoztatott cél-függvény és F fokusz pont.

A(z) 3. ábrán láthatjuk, milyen változásokhoz vezet a p_1 nem-bázis együtthatóban történő

$$p_1 = 6 \quad \longrightarrow \quad p_1' = p_1 + 1 = 6 + 1 = 7$$

változás.

2. Változás a jobb oldali korlátvektorban

Cseréljük ki a jobb oldali b vektor b_μ elemét a $b'_\mu = b_\mu + \delta$ értékre, és vizsgáljuk meg az adott csere hatását a B optimális bázisra, az x^* optimális megoldásra és a $Q(x)$ cél-függvény optimális értékére.

A korábban megtett feltételezésünk szerint az x^* vektor lehetséges és optimális megoldása a(z) (92)-(94) feladatnak. Mivel az x^* vektor lehetséges megoldása ennek a feladatnak, ezért a x^* vektor mellett teljesülnek a(z) (93) és (94) feltételek. Írjuk át a(z) (93) rendszert a következő módon: (93) as

follows

$$Bx^* = b, \tag{97}$$

vagy

$$x^* = B^{-1}b, \tag{98}$$

ahol $B^{-1} = ||e_{ij}||_{m\times m}$ jelöli a B mátrix inverzét (emlékezve arra, hogy B mátrix lineárisan független $A_j,\ j = 1, 2, \ldots, m,$ vektorokból áll, tehát B^{-1} létezik).

Továbbá a(z) (98) rendszerből kapjuk, hogy

$$x_i^* = \sum_{k=1}^{m} e_{ik}b_k,\ \ i = 1, 2, \ldots, m. \tag{99}$$

Vezessük be a $b_\mu \to b_\mu + \delta$ cserét a(z) (99) képletbe:

$$x_i' = \sum_{k=1}^{m} e_{ik}b_k + \delta e_{i\mu} = x_i^* + \delta e_{i\mu},\ \ i = 1, 2, \ldots, m. \tag{100}$$

Így az eredeti x^* vektor helyett kapjuk az

$$x' = (x_1', x_2', \ldots, x_m', 0, 0, \ldots, 0).$$

vektort.

Ezzel a vektorral kapcsolatban felvetődik a következő két kérdés:

(1) Mondhatjuk-e, hogy az x' vektor lehetséges megoldása az új feldatnak?
(2) Mondhatjuk-e, hogy az x' vektor optimális megoldása az új feldatnak?

Vizsgáljuk meg az első kérdést. A lehetséges megoldás definíciója szerint ahhoz, hogy az x' vektor lehetséges megoldása legyen a(z) (92)-(94) feladatnak, kell, hogy teljesüljön az

$$x_i' \geq 0,\ \ i = 1, 2, \ldots, m; \tag{101}$$

és

$$\begin{cases} \sum_{j=1}^{m} a_{ij}x_j' = b_i, & i = 1, 2, \ldots, m,\ \ i \neq \mu; \\ \sum_{j=1}^{m} a_{ij}x_j' = b_i + \delta, & i = \mu \end{cases} \tag{102}$$

feltétel. A(z) (100) használatával írjuk át (101)-t a közetkezőképpen:

$$x_i' = x_i^* + \delta e_{i\mu} \geq 0,\ \ i = 1, 2, \ldots, m.$$

Az utóbbi azt jelenti, hogy

$$\delta e_{i\mu} \geq -x_i^*,\ \ i = 1, 2, \ldots, m,$$

vagy

$$\delta \geq -\frac{x_i^*}{e_{i\mu}}, \quad \text{for those } i \text{ that } \ e_{i\mu} > 0,$$
$$\delta \leq -\frac{x_i^*}{e_{i\mu}}, \quad \text{for those } i \text{ that } \ e_{i\mu} < 0.$$

Ilyen módon a következő tartományt kaphatjuk:

$$\max_{\substack{1\leq i\leq m\\ e_{i\mu}>0}} \left\{-\frac{x_i^*}{e_{i\mu}}\right\} \leq \delta \leq \min_{\substack{1\leq i\leq m\\ e_{i\mu}<0}} \left\{-\frac{x_i^*}{e_{i\mu}}\right\}. \tag{103}$$

Nyilvánvaló, ha δ értéke olyan, hogy teljesül a(z) (103) feltétel, akkor minden $x_i', \ \ i = 1, 2, \ldots, m, \ \ $ bázisváltozó nem-negatív értékű, és ezért teljesül a(z) (101) feltétel.

Ami a(z) (102) feltételt illeti, az x' vektor definíció szerint teljesíti ezt a feltételt. Tényleg, az x' vektor meghatározásakor a B bázismátrixot és (97) képletet használtuk.

Ily módon megmutattuk, hogy ha δ teljesíti a(z) (103) feltételt, akkor az x' vektor mellett teljesül a(z) (101) és (102) feltétel, azaz az x' vektor lehetséges megoldása a módosított feladatnak.

Vizsgáljuk a második kérdést. A szimplex módszer optimalitási krítériuma szerint (lásd. 2. szekció) a lehetséges x' vektor akkor optimális vektor, ha

$$\Delta_j(x') = \Delta_j' - Q(x')\Delta_j'' \geq 0, \ \ j = 1, 2, \ldots, n.$$

Mivel

$$\Delta_j' \ = \ \Delta_j'' \ = \ \Delta_j(x^*) \ = \ 0, \quad \forall j \in J_B,$$

ezért az x' vektor optimalitásához elégséges, ha

$$\Delta_j(x') = \Delta_j' - Q(x')\Delta_j'' \geq 0, \ \ \forall j \in J_N. \tag{104}$$

Tekintsük a(z) (104) képletet. Jegyezzük meg, hogy a Δ_j' és Δ_j'' értékek közvetlenül nem függnek a b és x' vektoroktól. Ezért a jobb oldali b vektorban történő változások csak a $Q(x)$ cél-függvény optimális értékére hatnak. Tehát

$$Q(x') \quad = \quad \frac{P(x^*)}{D(x^*)} = \frac{\displaystyle\sum_{i=1}^{m} p_i x_i' + p_0}{\displaystyle\sum_{i=1}^{m} d_i x_i' + d_0} \ =$$

$$\overset{(100)}{=} \ \frac{\displaystyle\sum_{i=1}^{m} p_i(x_i^* + \delta e_{i\mu}) + p_0}{\displaystyle\sum_{i=1}^{m} d_i(x_i^* + \delta e_{i\mu}) + d_0} \ = \ \frac{P(x^*) + \delta h_1}{D(x^*) + \delta h_2}, \tag{105}$$

ahol

$$h_1 = \sum_{i=1}^{m} p_i e_{i\mu}, \quad h_2 = \sum_{i=1}^{m} d_i e_{i\mu}.$$

Ahhoz, hogy teljesüljön a $D(x) > 0$, $\forall x \in S$ feltétel, szükséges a

$$D(x^*) + \delta h_2 > 0. \tag{106}$$

feltétel teljesítése. Az utóbbi a következő δ tartományt adja:

$$\delta \begin{cases} > -\dfrac{D(x^*)}{h_2}, & \text{if} \quad h_2 > 0, \\ < -\dfrac{D(x^*)}{h_2}, & \text{if} \quad h_2 < 0. \end{cases} \tag{107}$$

A(z) (105) használatával továbbá átírhatjuk a(z) (104) feltételt a következő formába:

$$\Delta'_j \geq \Delta''_j \frac{P(x^*) + \delta h_1}{D(x^*) + \delta h_2}, \quad \forall j \in J_N. \tag{108}$$

A megfeleő átalakítások után kapjuk a

$$\delta(\Delta'_j h_2 - \Delta''_j h_1) \geq \Delta''_j P(x^*) - \Delta'_j D(x^*), \quad \forall j \in J_N,$$

feltételt, amelyből következik, hogy

$$\delta(\Delta'_j h_2 - \Delta''_j h_1) \geq -\Delta_j(x^*) D(x^*), \quad \forall j \in J_N.$$

Ebből kapjuk a következő tartományt:

$$\max_{\substack{j \in J_N \\ g_j > 0}} \left\{ \frac{-\Delta_j(x^*) D(x^*)}{g_j} \right\} \leq \delta \leq \min_{\substack{j \in J_N \\ g_j < 0}} \left\{ \frac{-\Delta_j(x^*) D(x^*)}{g_j} \right\}, \tag{109}$$

ahol

$$g_j = \Delta'_j h_2 - \Delta''_j h_1, \;\; j \in J_N.$$

Tehát ha δ teljesíti a(z) (109) és (107) feltételeket, akkor teljesül a(z) (104) feltételrendszer, és akkor az x' vektor a módosított hiperbolikus programozási feladatnak optimális megoldása.

Összefoglalás: *Ha δ kielégíti a(z) (103), (107) és (109) feltételeket, akkor a(z) (100) képletből meghatározható x' vektor optimális megoldása a módosított (azaz a $b_\mu \to b'_\mu = b_\mu + \delta$ csere után kapott) hiperbolikus programozási feladatnak.*

3. Változás a p számláló vektorban

Vizsgáljuk meg azt az esetet, amikor a $Q(x)$ cél-függvény számlálójában szereplő $p = (p_1, p_2, \ldots, p_n)$ vektor p_μ elemét kicseréljük a $p'_\mu = (p_\mu + \delta)$ elemre.

Feltételezhetjük, hogy az eredeti (92)-(94) feladat optimális megoldása az

$$x^* = (x_1^*, x_2^*, \ldots, x_m^*, 0, 0, \ldots, 0)^T.$$

vektor. Célunk az, hogy meghatározzuk a δ érték olyan alsó és felső határait, amelyek mellett a $p_\mu \to p'_\mu$ csere nem hat az optimális bázisra, és az x^* vektor marad az optimális megoldás.

Az ilyen csere hatásának vizsgálatakor a következő két esetet kell megkülönböztetnünk:

(1) $\mu \in J_N = \{m+1, m+2, \ldots, n\}$, azaz μ nem-bázis index;
(2) $\mu \in J_B = \{1, 2, \ldots, m\}$, azaz μ bázis index.

Nyilvánvaló, hogy a $p_\mu \to p'_\mu$ csere nem változtatja az S lehetséges halmazt, ezért az x^* vektor lehetséges megoldasa a módosított feladatnak. Azonban a p vektorban történő változás megváltoztatja a $Q(x)$ cél-függvény értékét, és ezzel változtatja a $\Delta_j(x^*) = \Delta'_j - Q(x^*)\,\Delta''_j$ determinánsokat is. Éppen emiatt a $p_\mu \to p'_\mu$ csere mellett előfordulhat, hogy az eredeti feladatban kapott x^* optimális megoldás az új feladatban már nem lesz optimális. Szóval, most ki kell derítenünk, hogy a $p_\mu \to p'_\mu$ csere hogyan hat a $\Delta_j(x^*),\ j = 1, 2, \ldots, n,$ determinánsokra.

1. Eset ($\mu \in J_N$): Ebben az esetben, mivel μ nem-bázis index, ezért $x_\mu^* = 0$ és $Q(x^*)$ értéke nem változik. Továbbá nem-bázis indexű p_μ együttható nem szerepel a bázis indexű $\Delta'_j,\ j = 1, 2, \ldots, m$, értékekben, és ezért a p_μ értékében történő változás nem befolyásolhatja a bázis $\Delta'_j,\ j = 1, 2, \ldots, m$, értéket. Azonban p_μ együttható egyetlen egy nem-bázis indexű

$$\Delta'_\mu = \sum_{i=1}^{m} p_i x_{i\mu} - p_\mu.$$

értékben sem szerepel. Ezért a $p_\mu \to p'_\mu$ csere esetén a Δ'_μ érték a következő módon változik:

$$\begin{aligned} \tilde{\Delta}'_\mu &= \sum_{i=1}^{m} p_i x_{i\mu} - p'_\mu = \sum_{i=1}^{m} p_i x_{i\mu} - (p_\mu + \delta) = \\ &= \sum_{i=1}^{m} p_i x_{i\mu} - p_\mu - \delta = \Delta'_\mu - \delta. \end{aligned}$$

Szóval,

$$\tilde{\Delta}_\mu(x^*) = (\Delta'_\mu - \delta) - Q(x^*)\Delta''_\mu = \Delta_\mu(x^*) - \delta.$$

Ebből adódik a következő feltétel:

$$\delta \leq \Delta_\mu(x^*). \tag{110}$$

Összefoglalás: *Ha a $\mu \in J_N$ és δ kielégíti a(z) (110) feltételt, akkor az eredeti feladat x^* optimális megoldása az új feladatnak is optimális megoldása.*

2. Eset ($\mu \in J_B$): Mivel μ a bázis index, ezért a $p_\mu \to p'_\mu$ csere megváltoztatja a $P(x)$ és $Q(x)$ függvények optimális értékét. Nyilvánvaló, hogy ilyenkor

$$\begin{aligned}\tilde{P}(x^*) &= \sum_{\substack{j\in J_B\\ j\neq\mu}} p_j x_j^* + p'_\mu x_\mu^* + p_0 = \sum_{\substack{j\in J_B\\ j\neq\mu}} p_j x_j^* + (p_\mu + \delta)x_\mu^* + p_0 = \\ &= \sum_{i=1}^{m} p_j x_j^* + p_0 + \delta x_\mu^* = P(x^*) + \delta x_\mu^*,\end{aligned}$$

és ezért

$$\tilde{Q}(x^*) = \frac{\tilde{P}(x^*)}{D(x^*)} = \frac{P(x^*) + \delta x_\mu^*}{D(x^*)}.$$

Ezenkívül a $p_\mu \to p'_\mu$ csere miatt változnak a nem-bázis indexű $\Delta'_j,\ j \in J_N$ értékek is:

$$\begin{aligned}\tilde{\Delta}'_j &= \sum_{\substack{i\in J_B\\ i\neq\mu}} p_i x_{ij} + p'_\mu x_{\mu j} - p_j = \sum_{\substack{i\in J_B\\ i\neq\mu}} p_i x_{ij} + (p_\mu + \delta)x_{\mu j} - p_j = \\ &= \sum_{i=1}^{m} p_i x_{ij} - p_j + \delta x_{\mu j} = \Delta'_j + \delta x_{\mu j},\ \ j \in J_N.\end{aligned}$$

Az előző képleteket figyelembe véve most meghatározhatjuk a nem-bázis indexű $\Delta_j(x^*),\ j \in J_N$ determinánsokat:

$$\begin{aligned}\tilde{\Delta}_j(x^*) &= \tilde{\Delta}'_j - \tilde{Q}(x^*)\Delta''_j = \\ &= \Delta'_j + \delta x_{\mu j} - \frac{P(x^*) + \delta x_\mu^*}{D(x^*)}\,\Delta''_j,\ \ j \in J_N.\end{aligned} \tag{111}$$

Ami a bázis indexeket $(j \in J_B)$ illeti, nyilvánvaló, hogy bázis indexű Δ'_j értékek nem változnak, ezért

$$\tilde{\Delta}'_j = \Delta'_j = 0, \quad \forall j \in J_B,$$

és emiatt

$$\tilde{\Delta}_j(x^*) = \Delta_j(x^*) = 0, \quad \forall j \in J_B.$$

A szimplex módszer elmélete szerint ha teljesül a

$$\tilde{\Delta}_j(x^*) \geq 0,\ \ \forall j \in J \tag{112}$$

feltétel, akkor az x^* vektor optimális megoldása az új feladatnak. Mivel

$$\Delta'_j \;=\; \Delta''_j \;=\; \Delta_j(x^*) \;=\; 0, \quad \forall j \in J_B,$$

ezért a(z) (112) feltételekből csak azokra kell figyelnünk, amelyknél a j index nem-bázis. Szóval, a(z) (112) helyett kapjunk a következőt:

$$\tilde{\Delta}_j(x^*) \geq 0, \;\; \forall j \in J_N.$$

Az utóbbiból a(z) (111) képlet felhasználásával kapjuk a következő rendszert:

$$\Delta'_j + \delta x_{\mu j} - \frac{P(x^*) + \delta x^*_\mu}{D(x^*)}\, \Delta''_j \geq 0, \;\; \forall j \in J_N.$$

Ebből a rendszerből adódik, hogy

$$\delta(x_{\mu j} D(x^*) - \Delta''_j x^*_\mu) \geq -(D(x^*)\Delta'_j - P(x^*)\Delta''_j), \;\; \forall j \in J_N,$$

vagy másképpen

$$\delta(x_{\mu j} D(x^*) - \Delta''_j x^*_\mu) \geq -\Delta_j(x^*) D(x^*), \;\; \forall j \in J_N.$$

Az utóbbiból kapjuk a következő tartományt:

$$\max_{\substack{j \in J_N \\ g_j > 0}} \left\{ \frac{-\Delta_j(x^*) D(x^*)}{g_j} \right\} \leq \delta \leq \min_{\substack{j \in J_N \\ g_j < 0}} \left\{ \frac{-\Delta_j(x^*) D(x^*)}{g_j} \right\}, \tag{113}$$

ahol

$$g_j = x_{\mu j} D(x^*) - \Delta''_j x^*_\mu, \quad \forall j \in J_N. \tag{114}$$

Összefoglalás: *Ha a $\mu \in J_B$ és δ kielégíti a(z) (113) feltételt, akkor az eredeti feladat x^* optimális megoldása az új feladatnak is optimális megoldása.*

4. Változás a p_0 számláló együtthatóban

Tekintsük a p_0 együtthatóban történő $p_0 \to p'_0$ változást. Mint ahog mindig, feltételezzük, hogy az eredeti feladat otimális megoldása az

$$x^* = (x^*_1, x^*_2, \ldots, x^*_m, 0, 0, \ldots, 0)^T$$

vektor.

Cseréljük ki a p_0 együtthatót $p'_0 = (p_0 + \delta)$ értékűre, és vizsgáljuk meg hatását ennek a cserének az optimális megoldásra.

Elsősorban azt jegyezz uk meg, hogy a $p_0 \to p'_0$ nem változtaja az S lehetséges halmazt, ezért tetszőleges $p_0 \to p'_0$ csere mellett az x^* vektor lesz a lehetséges megoldása az új feladatnak. Ugyanakkor a $p_0 \to p'_0$ csere miatt megváltozik a $P(x^*)$ érték, és emiatt változik $Q(x^*)$, az utóbbi pedig a $\Delta_j(x^*),\; j = 1, 2, \ldots, n$, értékek változásához vezet, és ezzel befolyásolhatja az x^* vektor optimalitását.

Határozzuk meg a $P(x^*)$, a $Q(x^*)$ és a $\Delta_j(x^*),\ j=1,2,\dots,n$, értékekben történő változásokat. Nyilvánvaló, hogy

$$\tilde{P}(x^*)=\sum_{j=1}^{n}p_jx_j^*+p_0'=\sum_{j=1}^{n}p_jx_j^*+p_0+\delta=P(x^*)+\delta,$$

$$\tilde{Q}(x^*)=\frac{\tilde{P}(x^*)}{D(x^*)}=\frac{P(x^*)+\delta}{D(x^*)}$$

és ezért

$$\tilde{\Delta}_j(x^*)=\Delta_j'-\tilde{Q}(x^*)\,\Delta_j''=\Delta_j'-\frac{P(x^*)+\delta}{D(x^*)}\,\Delta_j'',\quad j=1,2,\dots,n. \tag{115}$$

A szimplex módszer elmélete szerint ha teljesül a

$$\tilde{\Delta}_j(x^*)\geq 0,\ \ \forall j\in J=\{1,2,\dots,n\},$$

feltétel-rendszer, akkor az x^* vektor az új (a $p_0\to p_0'$ csere után kapott) feladatnak optimális megoldása.

Továbbá, mivel p_0 nem szerepel sem a $\Delta_j',\ j=1,2,\dots,n,$ értékekben, sem a $\Delta_j'',\ j=1,2,\dots,n,$ értékekben, ezért ezek az értékek nem változnak. Innen következik, hogy bázis indexű Δ_j' és Δ_j'' (azaz $\forall j\in J_B$) maradnak nulla értékűek, azaz

$$\tilde{\Delta}_j'\ =\ \tilde{\Delta}_j''\ =\ \Delta_j'\ =\ \Delta_j''\ =\ 0,\quad \forall j\in J_B,$$

és emiatt

$$\tilde{\Delta}_j(x^*)\ =\ \Delta_j'-\tilde{Q}(x^*)\,\Delta_j''\ =\ 0,\ \ \forall j\in J_B.$$

Szóval, ahhoz, hogy x^* vektor optimális megoldása legyen az új feladatnak, elégendő, ha

$$\tilde{\Delta}_j(x^*)\geq 0,\ \ \forall j\in J_N.$$

Innen a(z) (115) használatával kapjuk a következőt:

$$\Delta_j'-\frac{P(x^*)+\delta}{D(x^*)}\,\Delta_j''\geq 0,\ \ \forall j\in J_N.$$

Az utóbbi a megfelelő átalakítások után adja a

$$\delta\,\Delta_j''\leq\Delta_j'\,D(x^*)-P(x^*)\Delta_j',\quad \forall j\in J_N$$

feltételt, amelyből kapjuk, hogy

$$\max_{\substack{j\in J_N\\ \Delta_j''<0}}\left\{\frac{\Delta_j(x^*)D(x^*)}{\Delta_j''}\right\}\ \leq\ \delta\ \leq\ \min_{\substack{j\in J_N\\ \Delta_j''>0}}\left\{\frac{\Delta_j(x^*)D(x^*)}{\Delta_j''}\right\}. \tag{116}$$

Összefoglalás: *Ha δ kielégíti a(z) (116) feltételt, akkor az eredeti feladat x^* optimális megoldása az új feladatnak is optimális megoldása.*

5. Változás a d nevező vektorban

Ebben a részben vizsgálni fogjuk azt az esetet, amikor a $Q(x)$ cél-függvény $D(x)$ nevezőjében szereplő $d = (d_1, d_2, \ldots, d_n)$ vektor d_μ eleme változik.

Tegyük fel, hogy

$$x^* = (x_1^*, x_2^*, \ldots, x_m^*, 0, 0, \ldots, 0)^T$$

vektor az eredeti feladat optimális megoldása. Cseréljük ki a d_μ elemet a $d'_\mu = (d_\mu + \delta)$ értékre.

Jegyezzük meg, hogy a $d_\mu \to d'_\mu$ csere nem változtatja az S lehetséges megoldást, és ezért az x^* az új feladatnak lehetséges megoldása. Azonban valtozhat $D(x^*)$, változhat $Q(x^*)$ és ezért változhatnak a $\Delta_j(x^*)$ determinánsok. Szóval, a $d_\mu \to d'_\mu$ csere miatt kérdésessé válhat az x^* vektor optimalitása.

Tekintsük a következő két esetet:

– $\mu \in J_N = \{m+1, m+2, \ldots, n\}$, azaz μ nem-bázis index;
– $\mu \in J_B = \{1, 2, \ldots, m\}$, azaz μ bázis index.

Mindenekelőtt jegyezzük meg, hogy a $d_\mu \to d'_\mu$ csere nem változtatja az eredeti feladat S lehetséges halmazát, és ezért az x^* vektor az új feladatnak lehetséges megoldása lesz.

Ugyanakkor a $d_\mu \to d'_\mu$ csere miatt valtozhat (a μ indextől függően) a $D(x)$ és a $Q(x) = P(x)/D(x)$ függvény értéke. A utóbbi pedig megváltoztathatja a $\Delta_j(x^*) = \Delta'_j - Q(x^*)\Delta''_j$ determinánsokat, és ezzel befolyásolhatja az x^* vektor optimalitását.

Vizsgáljuk meg a $d_\mu \to d'_\mu$ csere hatását a $\Delta_j(x^*)$ determinánsokra.

1. Eset ($\mu \in J_N$): Mivel μ nem-bázis index, ezért $x_\mu^* = 0$ és $Q(x)$ optimális értéke nem változik. Jegyezzük meg, hogy ilyenkor $D(x)$ értéke sem változik, és ezért $D(x)$ megtartja az értékének pozitivitását.

Továbbá, mivel d_μ nem-bázis együttható, és nem szerepel bázis indexű Δ''_j, $j \in J_B$, értékekben, ezért a bázis indexű Δ''_j, $j = 1, 2, \ldots, m$, értékek nem változnak. Azonban a d_μ együttható szerepel nem-bázis Δ''_μ, értékben

$$\Delta''_\mu = \sum_{i=1}^{m} d_i x_{i\mu} - d_\mu$$

és ezért a $d_\mu \to d'_\mu$ csere esetén

$$\begin{aligned}\tilde{\Delta}''_\mu &= \sum_{i=1}^{m} d_i x_{i\mu} - d'_\mu = \sum_{i=1}^{m} d_i x_{i\mu} - (d_\mu + \delta) = \\ &= \sum_{i=1}^{m} d_i x_{i\mu} - d_\mu - \delta = \Delta''_\mu - \delta\end{aligned}$$

és emiatt

$$\begin{aligned}\tilde{\Delta}_\mu(x^*) &= \Delta'_\mu - Q(x^*)\tilde{\Delta}''_\mu = \\ &= \Delta'_\mu - Q(x^*)(\Delta''_\mu - \delta) = \Delta_\mu(x^*) + Q(x^*)\,\delta.\end{aligned}$$

Az utóbbi azt jelenti, hogy ha δ teljesíti a

$$\Delta_\mu(x^*) + Q(x^*)\,\delta \geq 0 \tag{117}$$

feltételt, akkor az eredeti feledat x^* optimális megoldása optimális megoldása lesz az új feladatnak is. A(z) (117) feltétel használatával a következő tartományt kapjuk:

$$\delta \begin{cases} \geq \dfrac{-\Delta_\mu(x^*)}{Q(x^*)}, & \text{ha } Q(x^*) > 0, \\ \leq \dfrac{-\Delta_\mu(x^*)}{Q(x^*)}, & \text{ha } Q(x^*) < 0, \\ \text{korlátlan}, & \text{ha } Q(x^*) = 0. \end{cases} \tag{118}$$

Összefoglalás: *Ha $\mu \in J_N$ és δ kielégíti a(z) (118) feltételt, akkor az eredeti feladat x^* optimális megoldása az új feladatnak is optimális megoldása.*

2. Eset ($\mu \in J_B$): Ebben az esteben a $d_\mu \to d'_\mu$ csere megváltoztatja $D(x^*)$ értéket, mégpedig a következő módon:

$$\begin{aligned}\tilde{D}(x^*) &= \sum_{\substack{j\in J_B \\ j\neq\mu}} d_j x_j^* + d'_\mu x_\mu^* + d_0 = \sum_{\substack{j\in J_B \\ j\neq\mu}} d_j x_j^* + (d_\mu + \delta)x_\mu^* + d_0 = \\ &= D(x^*) + \delta x_\mu^*,\end{aligned}$$

és ezzel megváltoztatja a $Q(x^*)$ értékét is:

$$\tilde{Q}(x^*) = \frac{P(x^*)}{\tilde{D}(x^*)} = \frac{P(x^*)}{D(x^*) + \delta x_\mu^*}\ .$$

Ezenkívül, a $D(x)$ nevező nem lehet negatív vagy nulla értékű, ebből keletkezik még egy feltétel:

$$\tilde{D}(x^*) = D(x^*) + \delta x_\mu^* > 0,$$

amelyből kapjuk a következő korlátozást:

$$\delta > \frac{-D(x^*)}{x_\mu^*}. \tag{119}$$

A(z) (119) képletben feltételezzük, hogy az x^* vektor nem-degenerált, és ezért $x^*_\mu > 0$. Nyilvánvaló, ha x^* degenerált, és az éppen μ-edik bázis komponense nulla értékű, akkor a $d_\mu \to d'_\mu,\ \mu \in J_B,$ cserének nincsen semmi hatása a $D(x)$ függvény pozitivitására, és ezért δ értéke lehet korlátlan.

Továbbá a $d_\mu \to d'_\mu,\ \mu \in J_B,$ csere megváltoztatja a nem-bázis indexű $\Delta''_j,\ j \in J_N$, értékeket

$$\begin{aligned}\tilde{\Delta}''_j &= \sum_{\substack{i\in J_B\\ i\neq\mu}} d_i x_{ij} + d'_\mu x_{\mu j} - d_j = \sum_{\substack{i\in J_B\\ i\neq\mu}} d_i x_{ij} + (d_\mu+\delta)x_{\mu j} - d_j = \\ &= \sum_{i=1}^{m} d_i x_{ij} - d_j + \delta x_{\mu j} = \Delta''_j + \delta x_{\mu j},\ \ j \in J_N.\end{aligned}$$

de nem változtatja meg sem a bázis indexű Δ''_j értékeket, sem a bázis indexű $\Delta_j(x^*)$ determinánsokat, $(\forall j \in J_B)$. Ezért azok maradnak nulla értékűek, azaz

$$\tilde{\Delta}''_j = \Delta''_j = 0 \quad \text{és} \quad \tilde{\Delta}_j(x^*) = \Delta_j(x^*) = 0, \quad \forall j \in J_B. \tag{120}$$

Most pedig derítsük ki a változásokat a nem-bázis indexű $\Delta_j(x^*),\ j \in J_N$, determinánsokban:

$$\begin{aligned}\tilde{\Delta}_j(x^*) &= \Delta'_j - \tilde{Q}(x^*)\tilde{\Delta''}_j = \\ &= \Delta'_j - \frac{P(x^*)}{D(x^*)+\delta x^*_\mu}\,(\Delta''_j + \delta x_{\mu j}),\ \ j \in J_N. \qquad (121)\end{aligned}$$

A szimplex módszer elmélete szerint az új feladat x^* lehetséges bázismegoldása optimális, ha

$$\tilde{\Delta}_j(x^*) \geq 0,\ \ \forall j \in J = \{1, 2, \ldots, n\}.$$

Figyelembe véve a(z) (120) képleteket az utóbbiból a(z) (121) használatával kapjuk a következő optimalitási feltételt:

$$\Delta'_j - \frac{P(x^*)}{D(x^*)+\delta x^*_\mu}\,(\Delta''_j + \delta x_{\mu j}) \geq 0,\ \ \forall j \in J_N,$$

vagy

$$\Delta'_j \geq \frac{P(x^*)}{D(x^*)+\delta x^*_\mu}\,(\Delta''_j + \delta x_{\mu j}),\ \ \forall j \in J_N. \tag{122}$$

Továbbá írjuk át a(z) (122) feltételt a(z) (119) használatával a következő formába:

$$\delta\,(\Delta'_j x^*_\mu - P(x^*)\,x_{\mu j}) \geq -(D(x^*)\Delta'_j - P(x^*)\Delta''_j), \quad \forall j \in J_N,$$

vagy

$$\delta\,(\Delta'_j x^*_\mu - P(x^*)\,x_{\mu j}) \geq -\Delta_j(x^*)D(x^*), \quad \forall j \in J_N.$$

Az utóbbiból adódik a következő tartomány:

$$\max_{\substack{j\in J_N\\ g_j>0}}\left\{\frac{-\Delta_j(x^*)D(x^*)}{g_j}\right\}\le\delta\le\min_{\substack{j\in J_N\\ g_j<0}}\left\{\frac{-\Delta_j(x^*)D(x^*)}{g_j}\right\}, \tag{123}$$

ahol

$$g_j=\Delta'_jx^*_\mu-P(x^*)\,x_{\mu j},\quad\forall j\in J_N.$$

Összefoglalás: *Ha $\mu\in J_B$, az x^* vektor nem-degenerált, és δ kielégítí a(z) (119) és (123) feltételeket, akkor az eredeti feladat x^* optimális megoldása az új feladatnak is optimális megoldása.*
Abban az esetben, ha az x^* vektor degenerált, és éppen $x^*_\mu=0$, akkor a(z) (119) feltétel teljesítése nem szükséges.

6. Változás a d_0 nevező együtthatóban

Ebben a szekcióban tekintsük a $d_0\to d'_0=(d_0+\delta)$ cserét.

Tegyük fel, hogy

$$x^*=(x^*_1,x^*_2,\dots,x^*_m,0,0,\dots,0)^T$$

vektor a(z) (92)-(94) eredeti hiperbolikus programozási feladat optimális megoldása a $B=(A_1,A_2,\dots,A_m)$ bázissal.

Jegyezzük meg, hogy a $d_0\to d'_0$ csere nem változtatja meg a feladat lehetséges halmazát, és ezért az eredeti hiperbolikus programozási feladat x^* optimális bázismegoldása lesz a lehetséges bázismegoldása az uj feldatnak is.

Azonban a $d_0\to d'_0$ csere megváltoztatja a $D(x)$ és a $Q(x)=P(x)/D(x)$ függvény optimális értékét, és ezúttal megváltoztathatja a

$$\Delta_j(x^*)=\Delta'_j-Q(x^*)\Delta''_j,\ \ j=1,2,\dots,n,$$

determinánsokat. Más szavakkal bármely változás a d_0 együttható értékében az x^* vektor optimalitásának elvesztéséhez vezethet.

Nyilvánvaló, hogy a $d_0\to d'_0$ cserénél a $Q(x)$ cél-függvény $D(x)$ nevezőjének értéke az x^* pontban a következő módon változik:

$$\tilde D(x^*)=\sum_{j=1}^n d_jx^*_j+d'_0=\sum_{j=1}^n d_jx^*_j+d_0+\delta=D(x^*)+\delta.$$

Ebből kapjuk, hogy

$$\tilde Q(x^*)=\frac{P(x^*)}{\tilde D(x^*)}=\frac{P(x^*)}{D(x^*)+\delta}$$

és ezért

$$\tilde{\Delta}_j(x^*) = \Delta'_j - \tilde{Q}(x^*)\,\Delta''_j = \Delta'_j - \frac{P(x^*)}{D(x^*)+\delta}\,\Delta''_j,\ \ j = 1,2,\ldots,n. \tag{124}$$

A kapott $\tilde{\Delta}_j(x^*)$ értékek használatával azonnal kiderül, hogy az adott δ mellett x^* optimális-e? Csak figyelembe kell venni, hogy a $D(x)$ függvény nem lehet negatív vagy nulla értékű, ezért δ értéke csak olyan lehet, hogy $D(x^*)+\delta > 0$. Ebből kapjuk az első feltételt:

$$\delta > -D(x^*). \tag{125}$$

Továbbá a szimplex módszer elmélete szerint az x^* vektor optimális megoldása az új feldatnak, ha

$$\tilde{\Delta}_j(x^*) \geq 0,\ \ \forall j \in J = \{1,2,\ldots,n\},$$

Vegyünk észre, hogy a d_0 együttható nem szerepel se Δ'_j, se Δ''_j, $j = 1,2,\ldots,n$, értékekben. Ez azt jelenti, hogy a bázis indexű Δ'_j és Δ''_j értékek megtartják a nulla értéket az új feladatban is, azaz

$$\tilde{\Delta}'_j \;=\; \tilde{\Delta}''_j \;=\; \Delta'_j \;=\; \Delta''_j \;=\; 0, \quad \forall j \in J_B.$$

Ebből következik, hogy

$$\tilde{\Delta}_j(x^*) \;=\; \tilde{\Delta}'_j - \tilde{Q}(x^*)\,\tilde{\Delta}''_j \;=\; 0 - \tilde{Q}(x^*)\,0 \;=\; 0,\ \ \forall j \in J_B.$$

Tehát a $d_0 \to d'_0$ csere megváltoztathatja a csak nem-bázis indexű $\Delta_j(x^*),\ j \in J_N$, determinánsokat, ezért ahhoz, hogy az x^* vektor optimális megoldása legyen az új feladatnak, elegendő, ha teljesülnek a

$$\tilde{\Delta}_j(x^*) \geq 0,\ \ \forall j \in J_N,$$

feltételek.

Ily módon a(z) (124) használatával kapjuk a következő feltételeket:

$$\Delta'_j - \frac{P(x^*)}{D(x^*)+\delta}\,\Delta''_j \geq 0,\ \ \forall j \in J_N. \tag{126}$$

A(z) (125) képlet figyelembe vételével az utóbbit átírhatjuk a következő formába:

$$\delta\Delta'_j \geq -D(x^*)(\Delta'_j - Q(x^*)\Delta''_j),\ \ \forall j \in J_N,$$

vagy

$$\delta\Delta'_j \geq -D(x^*)\Delta_j(x^*), \quad \forall j \in J_N.$$

Az alsó és felső korlátok végleges formája a következő:

$$\max_{\substack{j\in J_N\\ \Delta'_j>0}} \left\{ \frac{-\Delta_j(x^*)D(x^*)}{\Delta'_j} \right\} \;\leq\; \delta \;\leq\; \min_{\substack{j\in J_N\\ \Delta'_j<0}} \left\{ \frac{-\Delta_j(x^*)D(x^*)}{\Delta'_j} \right\}. \tag{127}$$

Összefoglalás: *A kapott eredményt megfogalmazhatjuk a következőként: ha a δ érték teljesíti a(z) (125) és (127) feltételeket, akkor az eredeti hiperbolikus programozási feladat x^* optimális megoldása az új feldatanak is optimális megoldása.*

V. Fejezet
Szállítási Feladat

Ebben a fejezetben egy speciális hiperbolikus programozási feladattal fogunk foglalkozni. 1941-ben F.L.Hitchcok vetette fel az azóta szállítási feladat néven ismert feladatot a lineáris programozásban. Később a lineáris programozási szállítási feladatot általánosították a hiperbolikus programozásra. Ennek a feladatnak több speciális változata van, pl. *hozzárendelési* (ang.–"*assignment problem*") feladat, *átrakodásos szállítási* (ang.–"*transshipment problem*") feladat. Annak ellenére, hogy ezeket a speciális struktúrájú feladatokat meg lehet oldani általános szimplex módszerrel, gyakran érdemes alkalmazni az éppen ezekre a feladatokra kifejlesztett módszereket.

Ebben a fejezeteben ezekkel a feladatokkal foglalkozunk. Ennek a fejezetnek a felépítése a következő: először megfogalmazzuk a feladatot, majd bevezetjük a megfelelő definiciókat és jelöléseket, majd áttekintjünk a speciális megoldási módszereket, a végén pedig megoldunk egy numerikus példát.

1. A feladat megfogalmazása

A szállítási feladat megfogalmazásában a következő adatok szerepelnek:

1.: m darab R_i-vel jelölt "feladóhely" (ang.–*"supply points"* vagy *"stores"*), vagy "raktár", ahol azonos anyagféleség (termékféleség) áll rendelkezésre különböző b_i mennyiségekben ("készlet") (ang.–*"supply"*), $i = 1, 2, \ldots, m$.

2.: n darab B_j-vel jelölt "felvevőhely" (ang. – *demand points* vagy *shops*) vagy "bolt", amelyeken az adott anyagféleségre (termékféleségre) van szükség adott a_j mennyiségben ("kereslet") (ang. – *"demand"*), $j = 1, 2, \ldots, n$.

3.:

$$P = ||p_{ij}||_{m\times n} = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mn} \end{pmatrix}$$

profit mátrix, amelynek p_{ij} eleme a profit, ha a szóban forgó anyag egy egységének szállítása az i-edik raktárról a j-edik boltba történik.

4.:

$$D = ||d_{ij}||_{m\times n} = \begin{pmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ d_{21} & d_{22} & \dots & d_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ d_{m1} & d_{m2} & \dots & d_{mn} \end{pmatrix}$$

költség mátrix, amelynek d_{ij} eleme a költség, ha a szóban forgó anyag egy egységének szállítása az i-edik raktárról a j-edik boltba történik.

5.: p_0 és d_0 állandók – a szállítástól nem függő állandó profit és költség.

Vezessünk be a következő ismeretlen változókat: x_{ij} – az i-edik raktárról a j-edik boltba szállítandó anyagmennyiség, $i = 1, 2, \dots, m,\ \ j = 1, 2, \dots, n$.

Az ilyen módon bevezetett jelölésekkel a probléma az alábbi hiperbolikus programozási feladattal írható le:

Adott a

$$Q(x) = \frac{P(x)}{D(x)} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}x_{ij} + p_0}{\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij}x_{ij} + d_0}, \tag{128}$$

cél-függvény, amelyet maximálizálni kell a következő feltételek mellett

$$\sum_{j=1}^{n} x_{ij} \le b_i, \quad i = 1, 2, \dots, m, \tag{129}$$

$$\sum_{i=1}^{m} x_{ij} \ge a_j, \quad j = 1, 2, \dots, n, \tag{130}$$

$$x_{ij} \ge 0, \quad i = 1, 2, \dots, m;\ j = 1, 2, \dots, n. \tag{131}$$

Itt és mindenütt a továbbiakban feltételezzük, hogy $D(x) > 0,\ \ \forall x = (x_{ij}) \in S,$ ahol S jelöli a(z) (129)-(131) feltételek által meghatározott lehetséges halmazt.

Ezenkívül feltételezzük, hogy

$$b_i > 0,\ \ a_j > 0,\ \ i = 1, 2, \dots, m;\ j = 1, 2, \dots, n, \tag{132}$$

és az összes készlet legalább olyan nagy, mint az összes kereslet, azaz

$$\sum_{i=1}^{m} b_i \geq \sum_{j=1}^{n} a_j. \tag{133}$$

Az ilyen módon megfogalmazott feladat rendelkezik néhány fontos tulajdonsággal:

– A feladat lehetséges halmaza nem üres, azaz $S \neq \emptyset$.
– A feladat lehetséges halmaza mindig korlátos.
– Innen következik, hogy a feladat mindig megoldható.

Tényleg, legyen

$$x'_{ij} = \frac{b_i a_j}{K}, \quad i = 1, 2, \ldots, m, \; j = 1, 2, \ldots, n, \tag{134}$$

ahol

$$K = \sum_{j=1}^{n} a_j > 0.$$

Az x'_{ij} értékeket behelyettesítve a(z) (129) és a(z) (130) feltételekbe a következőt kapjuk:

$$\sum_{j=1}^{n} x'_{ij} = \sum_{j=1}^{n} \frac{b_i a_j}{K} = \frac{b_i}{K} \sum_{j=1}^{n} a_j = \frac{b_i}{K} K = b_i, \quad i = 1, 2, \ldots, m,$$

és

$$\begin{aligned} \sum_{i=1}^{m} x'_{ij} &= \sum_{i=1}^{m} \frac{b_i a_j}{K} = \frac{a_j}{K} \sum_{i=1}^{m} b_i \overset{(133)}{\geq} \frac{a_j}{K} \sum_{j=1}^{n} a_j = \\ &= \frac{a_j}{K} K = a_j, \quad j = 1, 2, \ldots, n. \end{aligned}$$

Ez pedig azt jelenti, hogy az x'_{ij} értékek kielégítik a(z) (129) és (130) feltételeket.

A(z) (132) és a(z) (134) képletekből nyilvánvalóvá válik, hogy $x'_{ij} > 0$, $i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$.

Tehát az x'_{ij} értékek kielégítik minden feltételét a szállítási feladatnak. Ezzel megmutattuk, hogy a feladat lehetséges halmaza nem üres.

Továbbá a(z) (129) és (131)-ból következik, hogy

$$0 \leq x_{ij} \leq b_i, \quad i = 1, 2, \ldots, m; \; j = 1, 2, \ldots, n.$$

Ebből pedig következik, hogy S korlátos.

Végül, mivel $P(x)$ és $D(x)$ lineáris függvények, még ezenkívül $D(x) > 0$, $\forall x \in S$, ezért a $Q(x)$ függvény korlátos, és ezért az adott szállítási feladat megoldható.

V.1. Definíció. *Ha összes készlet pontosan megegyezik az összes kereslettel, azaz*

$$\sum_{i=1}^{m} b_i = \sum_{j=1}^{n} a_j, \tag{135}$$

akkor a feladatot ***egyensúlyozott*** *feladatnak nevezzük (angolul –* ***balanced transportation problem****, oroszul – "сбалансированная транспортная задача"). Egyébként a feladatot* ***nyitott*** *feladatnak szokták nevezni (angolul –* ***open transportation problem*** *vagy* ***un-balanced transportation problem****, oroszul – "несбалансированная транспортная задача" vagy "открытая транспортная задача").*

A(z) (128)-(131) szállítási feladatot néha még *nem-korlátos szállítási feladatnak* is nevezik, mivel a feldatban nincsenek megadva közvetlen formában az x_{ij} változókhoz tartozó felső korlátok. Meg kell jegyeznünk, hogy a(z) (129) és a(z) (131) feltételek mégis magukban foglalnak (de nem közvetlen módon!) felső korlátokat az x_{ij} változókra:

$$x_{ij} \leq \min_{1 \leq i \leq m} \{b_i\},\ i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n.$$

Ha a szállítási feladatban az összes készlet szigorúan kisebb az összes keresletnél, elvileg a feladat nem megoldható. Azonban ilyenkor az ún. *fiktív raktár* (ang.–*fictive supply point* vagy *dummy supply point*) bevezetésével a feladatot könnyen át lehet alakítani az egyensúlyozott formába. Ezenkívül a feladatban előfordulhat, hogy az összes készlet szigorúan nagyobb az összes keresletnél. Természtesen az ilyen feladat megoldható, de az ilyen feladatnak az egyensúlyozott formába való átalakításakor ún. *fiktív boltot* (ang.–*fictive demand point*) szoktak használni. Ezeknek a technikáknak részletes leírását megtalálhatjuk például a következő könyvekben [**80**], [**190**].

2. Hurokszerkesztéses Szimplex Módszer

Tekintsük a következő kanonikus szállítási feladatot:

$$Q(x) = \frac{P(x)}{D(x)} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} x_{ij} + p_0}{\sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij} + d_0} \longrightarrow \max \tag{136}$$

a következő feltételek mellett

$$\sum_{j=1}^{n} x_{ij} = b_i, \quad i = 1, 2, \ldots, m, \tag{137}$$

$$\sum_{i=1}^{m} x_{ij} = a_j, \quad j = 1, 2, \ldots, n, \tag{138}$$

$$x_{ij} \geq 0, \quad i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n, \tag{139}$$

ahol $D(x) > 0, \ \forall x = (x_{ij}) \in S$.

A(z) (137)-(138) feltételeknek megfelel a következő mátrix:

$$A\,|R = \left(\begin{array}{cccccccccccc|c}
1 & 1 & \cdots & 1 & & & & & & & & & b_1 \\
 & & & & 1 & 1 & \cdots & 1 & & & & & b_2 \\
 & & & & & & & & \ddots & & & & \vdots \\
 & & & & & & & & & 1 & 1 & \cdots\ 1 & b_m \\
1 & & & & 1 & & & & & 1 & & & a_1 \\
 & 1 & & & & 1 & & & & & 1 & & a_2 \\
 & & \ddots & & & & \ddots & & \cdots & & & \ddots & \vdots \\
 & & & 1 & & & & 1 & & & & 1 & a_n
\end{array}\right),$$

ahol R-rel jelöltük a b_i készleteket és az a_j keresleteket tartalmazó

$$R = (b_1, b_2, \ldots, b_m, a_1, a_2, \ldots, a_n)^T,$$

vektort.

Jelölje $A_{ij},\ i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n,$ az A ($m + n$ sorból és $m \times n$ oszlopból álló) mátrix oszlopait. Nyilvánvaló, hogy az A_{ij} oszlop-vektor '1'-t tartalmaz az i-edik és az $(m+j)$-edik pozicióban. Minden más eleme egyenlő nullával.

V.1. Tétel (Redundancia). *A(z) (137)-(138) feltétel-rendszerben van egyetlen egy redundáns feltétel. Ha az adott feltétel-rendszerből eltávolítunk tetszőlegesen egy feltételt, akkor a többi feltétellel olyan lineárisan független rendszert alkotnak, amelynek a rangja* $m + n - 1$.

Az állítás bizonyítása megtalálható K.G.Murty [**137**] könyvében.

V.2. Definíció. *Tetszőleges* $m + n - 1$ *darab* A_{ij} *vektorból álló (lineárisan független) B rendszert* ***bázisnak*** *fogunk nevezni.*

Például a

$$B = (A_{11}, A_{12}, \ldots, A_{1n}, A_{21}, A_{22}, \ldots, A_{2,m-1})$$

rendszer nevezhető bázisnak, mert éppen $m+n-1$ darab A_{ij} vektor szerepel benne.

Válasszunk ki az A mátrixból $m+n-1$ darab A_{ij} vektorból álló B bázist. Jelöljük J_B-vel a kiválasztott vektorok (ij) indexpárjait. Ha J-vel jelöljük az összes lehetséges (ij) indexpárt, akkor a $J_N = J \setminus J_B$ index-halmaz jelöli azoknak az A_{ij} vektorok indexparjait, amelyek nincsenek a bázisban.

V.3. Definíció. *Azt fogjuk mondani, hogy az $x = (x_{ij})$ változó a(z) (136)-(139) feladat* ***bázismegoldása****, ha x kielégíti a*

$$\sum_{(ij)\in J_B} A_{ij}x_{ij} = R \quad \text{és} \quad x_{ij} = 0, \ \forall (ij) \in J_N.$$

feltételt.

A szokásos módon ha az x_{ij} változó (ij) indexe a J_B halmazban van (azaz $(ij) \in J_B$), akkor ezt a változót *bázisváltozónak* fogjuk nevezni.

V.4. Definíció. *Azt fogjuk mondani, hogy az x bázismegoldás* ***degenerált****, ha legalább egy báziseleme egyenlő nullával, azaz ha $\exists (ij) : \ (ij) \in J_B$, olyan, hogy $x_{ij} = 0$. Egyébként az x bázismegoldást* ***nem-degeneráltnak*** *nevezzük.*

V.5. Definíció. *Azt fogjuk mondani, hogy az $x = (x_{ij})$ bázismegoldás a(z) (136)-(139) feladat* ***lehetséges bázismegoldása****, ha minden x_{ij} báziseleme (azaz $(ij) \in J_B$,) nem-negatív.*

Aa általános szállítási feladattól eltérően előfordulhat, hogy a kanonikus szállítási feladat nem megoldható.

V.2. Tétel. *A(z) (136)-(139) kanonikus szállítási feladat megoldható akkor és csak akkor, ha teljesül a*

$$\sum_{i=1}^{m} b_i = \sum_{j=1}^{n} a_j. \tag{140}$$

egyensúlyi feltétel *(angolul –* ***balance equality****, oroszul – "условие баланса").*

A következő állítások további fontos tulajdonságait adják meg a kanonikus szállítási feladatnak.

V.3. Tétel (Egészértékűség). *Ha minden a_j és b_i együttható a(z) (136)-(139) feladatban pozitív és egész értékű, akkor tetszőleges bázismegoldása csak egész számokból áll.*
Tehát ha minden a_j és b_i együttható pozitív és egész értékű, és teljesül az egyensúlyi feltétel, akkor a feladatnak van egészértékű optimális megoldása x^*.

Vezessük be a következő speciális változókat:
a $P(x)$ függvényhez rendeljük hozzá az $u'_i, \ i = 1, 2, \ldots, m, \ \text{és} \ v'_j, \ j = 1, 2, \ldots, n,$ változókat,

a $D(x)$ függvényhez pedig rendeljük hozzá az u_i'', $i = 1, 2, \ldots, m$, és v_j'', $j = 1, 2, \ldots, n$, változókat. A v_j' és v_j'' változók index szerint megfelelnek a "boltoknak", az u_i' és u_i'' változók pedig index szerint megfelenek a "raktároknak".

A változók meghatározásához használjuk az

$$u_i' + v_j' = p_{ij}, \quad (ij) \in J_B\ , \tag{141}$$

és az

$$u_i'' + v_j'' = d_{ij}, \quad (ij) \in J_B\ . \tag{142}$$

egyenletrendszereket.

A továbbiakban az u_i', v_j', u_i'', és v_j'' változók használatával vezessük be a következő Δ_{ij}' és Δ_{ij}'' értékeket:

$$\left.\begin{array}{rcl} \Delta_{ij}' & = & u_i' + v_j' - p_{ij} \\ \Delta_{ij}'' & = & u_i'' + v_j'' - d_{ij} \end{array}\right\} \quad i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n. \tag{143}$$

Továbbá vezessük be az

$$U_i(x) \ = \ u_i' - Q(x)\, u_i'' \ = \ \begin{vmatrix} u_i' & Q(x) \\ u_i'' & 1 \end{vmatrix}\ , \quad i = 1, 2, \ldots, m,$$

és

$$V_j(x) = v_j' - Q(x)\, v_j'' \ = \ \begin{vmatrix} v_i' & Q(x) \\ v_i'' & 1 \end{vmatrix}\ , \quad j = 1, 2, \ldots, n,$$

determinánsokat, majd

$$Z_{ij}(x) = U_i(x) + V_j(x), \quad i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n,$$

és

$$C_{ij}(x) = p_{ij} - Q(x)\, d_{ij}, \quad i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n,$$

értékeket, végül a

$$\Delta_{ij}(x) = Z_{ij}(x) - C_{ij}(x), \quad i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n.$$

determinánsokat.

Könnyen belátható, hogy a $\Delta_{ij}(x)$ determinánsokat kifejezhetjük a következő módon is:

$$\Delta_{ij}(x) = \Delta_{ij}' - Q(x)\Delta_{ij}'', \quad i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n. \tag{144}$$

Az adott jelölések használatával fogalmazzuk meg a következő optimalitási kritériumot:

V.4. Tétel (Optimalitási kritérium). *A(z) (136)-(139) kanonikus szállítási feladat $x = (x_{ij})$ lehetséges bázismegoldása optimális, ha teljesülnek a*

$$\Delta_{ij}(x) \geq 0, \quad i = 1, 2, \ldots, m,\ j = 1, 2, \ldots, n. \tag{145}$$

feltételek.

A feladathoz tartozó adatokat megfelelő felépítésű szállítási szimplex táblázatban szokták tartani és kezelni (lásd. 1. ábra), ahol T_{ij} jelöli x_{ij} bá-

	Bolt 1	Bolt 2	...	Bolt n	Készlet
Raktár 1	p_{11} T_{11} d_{11}	p_{12} T_{12} d_{12}	...	p_{1n} T_{1n} d_{1n}	b_1
Raktár 2	p_{21} T_{21} d_{21}	p_{22} T_{22} d_{22}	...	p_{2n} T_{2n} d_{2n}	b_2
⋮	⋮	⋮	⋮	⋮	⋮
Raktár m	p_{m1} T_{m1} d_{m1}	p_{m2} T_{m2} d_{m2}	...	p_{mn} T_{mn} d_{mn}	b_m
Kereslet	a_1	a_2	...	a_n	

1. Táblázat. Szállítási szimplex táblázat.

zisváltozókat, ha $(ij) \in J_B$ vagy $\Delta_{ij}(x)$ determinánsokat, ha $(ij) \in J_N$. Tehát a táblázat különböző cellái a következő tartalmúak:

$$\boxed{\begin{array}{lll} p_{ij} & & \\ & x_{ij} & \\ d_{ij} & & \end{array}}, \qquad \forall (ij) \in J_B,$$

és

$$\boxed{\begin{array}{lll} p_{ij} & & \\ & \Delta_{ij}(x) & \\ d_{ij} & & \end{array}} \quad \text{vagy} \quad \boxed{\begin{array}{lll} p_{ij} & & \Delta'_{ij} \\ & \Delta_{ij}(x) & \\ d_{ij} & & \Delta''_{ij} \end{array}}, \qquad \forall (ij) \in J_N.$$

V.6. Definíció. *A* ***huroknak*** *(angolul –* ***circle*** *vagy* ***loop****, oroszul – "цикл") nevezzük az olyan rendezett* (ij) *indexpárokat tartalmazó indexhalmazt, amelyben van legalább négy elem, és ezek az index-elemek rendelkeznek a következő tulajdonságokkal:*

(1) *Tetszőleges egymást követő indexpárnak a szimplex táblában vagy azonos oszlop, vagy azonos sor felel meg.*
(2) *Minden sorban és minden oszlopban van legfeljebb kettő hurokhoz tartozó indexpár.*
(3) *A hurok utolsó indexpárja vagy sorban vagy oszlopban megegyezik a hurok első indexpárjával.*

Azok a hurkok, amelyeket majd használni fogunk, rendelkeznek még egy speciális tulajdonsággal:
A hurok első eleme, mondjuk (rk) *legyen nem-bázis, azaz* $(rk) \in J_N$, *az összes többi eleme meg legyen bázisban.* Ilyen hurok esetén az (rk) indexű cellát a *hurkot generáló* cellának szokták nevezni.

	1	2	3	4
1	→	↓		
2		→		↓
3				
4	↑			←

	1	2	3	4	5
1			→		↓
2	↓				←
3					
4	→		↑		

2. Táblázat. Szállítási feladat – hurokkal

A(z) 2. táblában ábrázolt példák tartamazzák a következő két hurkot:

$$(1,1) \to (1,2) \to (2,2) \to (2,4) \to (4,4) \to (4,1) \to (1,1)$$

és

$$(1,3) \to (1,5) \to (2,5) \to (2,1) \to (4,1) \to (4,3) \to (1,3).$$

A(z) 3. táblán feltüntetett példák nem tartalmaznak hurkokat. Valóban, a bal oldali példában a legfelső sor tartalmaz kettőnél több cellát, a jobb oldali példában feltüntetett útvonal pedig nem alkot hurkot azért, mert a tábla második és harmadik oszlopa csak egy-egy cellát tartalmaz.

	1	2	3	4	5
1	→		→		↓
2	↑				←
3					

	1	2	3
1	→	↓	
2			
3			
4	↑		←

3. Táblázat. Szállítási feladat – hurkok nélkül

Tegyük fel, hogy B egy bázist jelöl, és x jelöli ennek a bázisnak megfelelő bázismegoldást.

Vizsgáljuk meg az x vektort az optimalitás szempontjából. Először állítsuk össze a(z) (141) és a(z) (142) rendszereket, majd az összeállított egyeletrenszerek megoldásával határozzzuk meg az u'_i, v'_j, u''_i, és v''_j változók értékét. Jegyezzük meg, hogy a(z) (141) és a(z) (142) rendszer összesen $(m+n-1)+(m+n-1)$ egyenletet tartalmaz, a változók száma pedig $(m+n)+(m+n)$. Mivel ezeknek a rendszereknek több megoldása van, ezért minden rendszerben egy-egy változót szabadon választhatunk meg. Megegyezés szerint ilyenkor

$$u'_1 \;=\; 0, \quad u''_1 \;=\; 0$$

értékekkel szokták indítani a számolást. Kiszámítva ezeket a változókat állítsuk össze a Δ'_{ij}, Δ''_{ij} és $\Delta'_{ij}(x)$ értékeket minden nem-bázis cellára. Ilyenkor a következő két eset fordulhat elő:

(1) Minden nem-bázis $\Delta_{ij}(x)$, $\forall(ij) \in J_N$, nem-negatív.
Mivel $\Delta_{ij}(x) = 0$, $\forall(ij) \in J_B$, ez azt jelenti, hogy $\Delta_{ij}(x) \geq 0$, $\forall(ij) \in J$.
(2) A nem-bázis $\Delta_{ij}(x)$ értékek között van legalább egy negatív értékű, azaz
$$J_N^- = \{(ij) |\, (ij) \in J_N,\ \Delta_{ij}(x) < 0\} \neq \emptyset.$$

Az 1. esetben az optimalitási kritériumnak (lásd. V.4. tétel) megfelelően az aktuális x lehetséges bázismegoldás optimális. Eljutottunk az optimális megoldáshoz. Vége.

A 2. esetben a J_N^- index-halmazból ki kell választanunk egy (rk) indexpárt, majd az x_{rk} változót bevezetjük a bázisba a következő szabályok szerint:

– Először jelöljük az (rk) indexű cellát '+'-jellel, és ebből a cellából kiindulva építsünk fel hurkot úgy, hogy közben jelöljük '-' és '+' jellel felváltva a hurokhoz tartozó cellákat. A hurok elkészítése után meghatározhatjuk a

$$\theta = \min_{(ij) \in J_B^-} x_{ij} = x_{fq}, \tag{146}$$

értéket, ahol J_B^- jelöli azoknak a báziscelláknak az indexhalmazát, amelyek '-' jellel szerepelnek a hurokban.
– Az
$$x_{ij}(\theta) = \begin{cases} x_{ij} - \theta, & \text{ha } (ij) \in J_B^-, \\ x_{ij} + \theta, & \text{ha } (ij) \in J_B^+, \end{cases}$$
képlet használatával számoljuk ki az új bázishoz tartozó bázisváltozókat. Itt J_B^+ jelöli azoknak a báziscellák az indexhalmazát, amelyek '+' jellel szerepelnek a hurokban.
– Minden nem-bázis változó értéke marad nulla.
– Az új bázisba kerülő x_{rk} változó új értéke $x_{rk}(\theta) = \theta$.
– Az x_{fq} bázisváltozó kikerül a bázisból, így az új értéke $x_{fq}(\theta) = 0$.

Az $x(\theta)$ új bázismegoldás előállítását követően újra (már az új bázisban) kiszámolhatjuk az u'_i, v'_j, u''_i, v''_j változókat és Δ'_{ij}, Δ''_{ij} $\Delta'_{ij}(x)$ értékeket, amelyek segítségével megvizsgálhatjuk az aktuális lehetséges bázismegoldást az optimalitás szempontjából. Mivel a kanonikus egyensúlyozott szállítási feladat megoldható, és az S lehetséges halmazhoz tartozó bázismegoldások (csúcspontok) száma véges, ezért a fenti eljárás véges számú ismétlése után eljutunk az optimális megoldáshoz.

Az adott szekció befejezése előtt meg kell jegyeznünk, hogy a hurokszerkesztéses szimplex módszer használata során kiderülhet, hogy a feladat aktuális lehetséges bázismegoldása degenerált.

Tegyük fel, hogy a θ érték meghatározásakor kiderült, hogy a(z) (146) képletben minimális érték több cellában is elérhető, azaz

$$\theta = \min_{(ij)\in J_B^-} x_{ij} = x_{f_1q_1} = x_{f_2q_2} = \ldots = x_{f_hq_h}.$$

Az utóbbi azt jelenti, hogy az új bázisban szereplő bázisváltozók között lesznek nullaértékűek, azaz az új bázismegoldás degenerált.

Tegyük fel, hogy az x lehetséges bázismegoldás degenerált. Ilyenkor előfordulhat, hogy az új hurokban egy vagy több nulla értékű bázisváltozó szerepel '-' jellel. Nyilvánvaló, hogy emiatt a θ értéke szintén nulla lesz, azaz $\theta = 0$. Az utóbbi pedig azt jelenti, hogy a cél-függvény értéke ilyenkor nem változík. Az ilyen ciklizálás elkerülése érdekében ugyanazokat a speciális szabályokat használhatjuk, amelykről szó volt a(z) 9. álfejezetben.

3. Az Induló Lehetséges Bázismogoldás Előállítása

A hurokszerkesztéses szimplex leírásánál feltételeztük, hogy rendelkezésünkre áll egy lehetséges bázismegoldás. Előfordulhat, hogy a feladattal együtt adott egy induló lehetséges bázis és hozzá lehetséges bázismegoldás. Ebben az esetben nincs akadálya a hurokszerkesztéses szimplex "beindításának". Leggyakrabban azonban nem ismeretes a megoldandó feladatnak egyetlen lehetséges bázismegoldása sem.

Ebben az alfejezetben bemutatunk néhány speciális módszert (eljárást), amelyek segítségével meg lehet határozni egy lehetséges bázismegoldást.

3.1. Északnyugati Sarok Módszer

Az adott eljárás nem használja a megoldandó feladathoz tartozó se $P = ||p_{ij}||_{m\times n}$ profit mátrixot, se $D = ||d_{ij}||_{m\times n}$ költség mátrixot.

Az adott módszer szerint a szimplex tábla balfelső ("északnyugati") celláját kell indulnunk – itt kell kiválasztanunk az x_{11} bázisváltozó értékét az

$$x_{11} \;=\; \min\{b_1, a_1\}$$

képlet alapján.
Ha $x_{11} = b_1$, akkor át kell húznunk (vagy lehet $\times$ jellel is jelölni) a tábla

legfelső sorát ezzel jelölve azt, hogy az első raktár készlete elfogyott, és

$$x_{12} = x_{13} = \ldots = x_{1n} = 0.$$

Ezután írjuk át az első bolt a_1 igényét az $a_1 - b_1$ értékre ezzel jelölve, hogy az első bolt az a_1 egységnyi igényéből b_1 egységet már kielégítettünk az $x_{11} = b_1$ szállítással.
Ha $x_{11} = a_1$, akkor át kell húznunk (vagy lehet $\times$ jellel is jelölni) a tábla bal oldali szélső oszlopát, ezzel jelölve azt, hogy az első bolt igényét teljesen kielégítettük, és

$$x_{21} = x_{31} = \ldots = x_{m1} = 0.$$

Ezután írjuk át az első raktár b_1 készletét a $b_1 - a_1$ értékre ezzel jelölve azt, hogy az első raktár a b_1 egységnyi készletéből a_1 egységet már kiszállítottunk az $x_{11} = b_1$ szállítással.
Ha $x_{11} = a_1 = b_1$, akkor tetszés szerint át kell huznunk vagy az első sort vagy az első oszlopot nem felejtve el megváltoztani a megfelelő módon a b_1 és a_1 értéket.

Ha az x_{11} változóval befejeztük a munkát, akkor a fenti eljárást alkalmazzuk a balfelső cellára a szimplex táblázat nem áthuzott részében. Az ehhez a cellához tartozó x_ij változó lesz a következő bázisváltozó.

Ezt a eljárást kell ismételnünk addig, amíg nem érjük el a tábla jobbalsó celláját.

	1	2	3	4	
1					200
2					100
3					100
4					100
	120	150	180	50	

4. Táblázat. Északnyugati sarok módszer – Eredeti tábla.

Illusztráljuk a módszer működését a(z) 4. táblázatban adott numerikus példán. Mivel az északnyugati sarok módszerben nincs szükségünk a célfüggvényben szereplő P mátrixra és D mátrixra egyikére sem, ezért ebben a példában nem használjuk ezeket az adatokat.

Kezdjük azzal, hogy kiválasztjuk az x_{11} változó értékét:

$$x_{11} = \min\{b_1, a_1\} = \min\{200, 120\} = 120.$$

Majd huzzuk át az első oszlopot, és változtassuk meg a b_1 és a_1 értéket (5. táblázat, 1. tábla):

$$b_1 \rightarrow b_1 - a_1 = 200 - 120 = 80$$
$$a_1 \rightarrow 0$$

	1	2	3	4	
1	120				80
2					100
3					100
4					100
	×	150	180	50	

	1	2	3	4	
1	120	80			×
2					100
3					100
4					100
	×	70	180	50	

5. Táblázat. Északnyugati sarok módszer – 1. és 2. tábla

	1	2	3	4	
1	120	80			×
2		70			30
3					100
4					100
	×	×	180	50	

	1	2	3	4	
1	120	80			×
2		70	30		×
3					100
4					100
	×	×	150	50	

6. Táblázat. Északnyugati sarok módszer – 3. és 4. tábla

	1	2	3	4	
1	120	80			×
2		70	30		×
3			100		×
4					100
	×	×	50	50	

	1	2	3	4	
1	120	80			×
2		70	30		×
3			100		×
4			50		50
	×	×	×	50	

7. Táblázat. Északnyugati sarok módszer – 5. és 6. tábla

	1	2	3	4	
1	120	80			×
2		70	30		×
3			100		×
4			50	50	×
	×	×	×	×	

8. Táblázat. Északnyugati sarok módszer – 7. tábla

Most következik az első oszlop áthúzása után megmaradt, át nem huzott táblarész észknyugati sarkában található x_{12} változó. Válasszuk az értékét:

$$x_{12} = \min\{b_1, a_2\} = \min\{80, 150\} = 80$$

Majd húzzuk át az első sort, és változtassuk a b_1 és a_2 értéket (5. táblázat, 2. tábla):

$$\begin{array}{rcl} a_2 \;\rightarrow\; a_2 - b_1 &=& 150 - 80 \;=\; 70 \\ b_1 \;\rightarrow\; 0 & & \end{array}$$

Folytassuk ezt a folyamatot amíg nem kapjuk a(z) 8. táblázatban feltüntetett végső 7. táblát. Így a következő lehetséges megoldást kapjuk:

$x_{11} = 120,\ \ x_{12} = 80,\ \ x_{22} = 70,\ \ x_{23} = 30,\ \ x_{33} = 100,\ \ x_{34} = 50,\ \ x_{44} = 50,$

a

$$J_B = \{(1,1),(1,2),(2,2),(2,3),(3,3),(4,3),(4,4)\},$$

bázis indexhalmazzal.

Jegyezzük meg, hogy az adott lehetséges megoldás pontosan $m+n-1 = 4+4-1 = 7$ nem-negatív változóból áll, és ezért az adott megoldás nem csak lehetséges, de még bázismegoldás is.

Ezenkívül figyelembe kell vennünk azt is, hogy a módszer nem használja a cél-függvényben szereplő p_{ij} és d_{ij} együtthatók egyikét sem, ezért gyakran olyan induló lehetséges bázismegoldást eredményez, amely viszonylag messze van az optimálistól.

A következő módszerben pedig van módunk a cél-függvény hasznosítására.

3.2. Maximális profit (vagy minimális költség) módszer

Az adott módszer tulajdonképpen a lineáris programozásból jól ismert minimális költség módszer olyan leszármazottja, amelyet két változatban használhatunk – az egyikben a $P = ||p_{ij}||_{m\times n}$ mátrixot használjuk (maximális profit módszer), a másikban pedig a $D = ||d_{ij}||_{m\times n}$ mátrixot (minimális költség módszer).

Mivel a szóban forgó két változat csak abban különbözik egymástól, hogy a bázisváltozó kiválasztásakor melyik mátrixot (P vagy D) kell figyelembe venni, ezért a módszer bemutatását csak a "profitos" változatára korlátozzuk.

A módszer azzal kezdödik, hogy a $P = ||p_{ij}||_{m\times n}$ mátrixból ki kell választani a legnagyobb értékű elemet:

$$p_{i_1 j_1} \;=\; \max p_{ij}$$

és a táblázat (i_1, j_1) cellájába be kell vezetni a legnagyobb lehetséges szállítást, azaz:

$$x_{i_1 j_1} \;=\; \min\{b_{i_1},\, a_{j_1}\}$$

Ugyanúgy, mint az északnyugati sarok módszer esetén, át kell húznunk vagy az i_1-edik sort vagy a j_1-edik oszlopot attól függően, hogy a b_{i_1} és az a_{j_1} közül melyik kisebb.

Majd az eljárást ismételnünk kell a táblázat át nem húzott részében addig, amíg egycellás át nem húzott táblázatot nem kapunk.

Tekintsük a 9. táblázatban megadott szállítási feladatot. Válaszzuk a P

	1	2	3	4	
1	8	6	6	1	200
2	3	4	6	8	100
3	7	3	8	9	100
4	4	12	4	3	100
	120	150	180	50	

9. Táblázat. Maximális profit módszer – Eredeti tábla, P mátrix.

mátrix legnagyobb elemét $\max p_{ij} = p_{4,2}$ és vezessünk be szállítást ebbe a cellába:

$$x_{42} = \min\{b_4, a_2\} = \min\{100, 150\} = 100.$$

Majd változtassuk $a_2 \to a_2 - b_4 = 50$, és húzzuk át a 4.-edik sort. A kapott eredményt írjuk be a táblába (lásd. 10. táblázat). A következő legnagyobb p_{ij} elemet tartalmazó cella $(3, 4)$. Tehát a következő bázisváltozó x_{34}. Határozzuk meg ennek a változónak az értékét

$$x_{34} = \min\{b_3, a_4\} = \min\{100, 50\} = 50,$$

majd változtassuk megfelelő módon a b_3 készletet és a_4 keresletet

$$b_3 \to b_3 - a_4 = 100 - 50 = 50$$
$$a_4 \to 0$$

és húzzuk át a 4.-edik oszlopot. Az eredményt írjuk be a 11. táblázatba. A következő lépésnél választhatjuk vagy az x_{11} változót vagy x_{33}-t, mert $p_{11} = p_{33} = 8$. Tetszés szerint válasszuk x_{11}-t, és a megfelelő átalakitások után kapjuk a 12. táblázatban feltüntetett eredményt. Utána következik az x_{33} változó (az eredmény megtekinthető a 13. táblázatban), majd következnek az $x_{12} = 50$, $x_{13} = 30$, $x_{23} = 100$ változók (az utolsó tábla megtekinthető a 14. táblázatban). Az ilyen módon kapott

	1	2	3	4	
1	8	6	6	1	200
2	3	4	6	8	100
3	7	3	8	9	100
4	4	12 **100**	4	3	×
	120	50	180	50	

10. Táblázat. Maximális profit módszer – 1. tábla

	1	2	3	4	
1	8	6	6	1	200
2	3	4	6	8	100
3	7	3	8	9 **50**	50
4	4	12 **100**	4	3	×
	120	50	180	×	

11. Táblázat. Maximális profit módszer – 2. tábla

$$x_{11} = 120,\ x_{12} = 50,\ x_{13} = 30,\ x_{23} = 100,\ x_{33} = 50,\ x_{34} = 50,\ x_{42} = 100$$

lehetséhes bázismegoldásnak megfelel a következő profitérték: $P(x) = 4090$.

Jegyezzük meg, hogy az északnyugati sarok módszer segítségével kapott lehetséges bázismegoldásnak csak a $P(x) = 3050$ érétkű profit felel meg.

	1	2	3	4	
1	8 **120**	6	6	1	80
2	3	4	6	8	100
3	7	3	8	9 **50**	50
4	4	12 **100**	4	3	×
	×	50	180	×	

12. Táblázat. Maximális profit módszer – 3. tábla

	1	2	3	4	
1	8 **120**	6	6	1	80
2	3	4	6	8	100
3	7	3	8 **50**	9 **50**	×
4	4	12 **100**	4	3	×
	×	50	130	×	

13. Táblázat. Maximális profit módszer – 4.tábla

3.3. Vogel-módszer

Ugyanúgy, mint az előző módszer esetén, a Vogel-módszer alkalmazása során egyaránt használhatjuk a P vagy a D mártixot. Mivel a módszer vagy sorokhoz, vagy oszlopokhoz alkalmazható, ezért beszélhetünk a módszer négy változatáról. Írjuk le a módszer lényegét a D mátrix és a táblazat oszlopainak használatával.

	1	2	3	4	
1	8 **120**	6 **50**	6 **30**	1	×
2	3	4	6 **100**	8	×
3	7	3	8 **50**	9 **50**	×
4	4	12 **100**	4	3	×
	×	×	×	×	

14. Táblázat. Maximális profit módszer – Végső tábla.

A módszer lényege az, hogy a táblázat minden oszlopához hozzá kell rendelni az ún. "*büntetéseket*" (ang.–"*penalty*"). A j-edik oszlophoz ($j = 1, 2, \ldots, n$) tartozó t_j büntetés egyenlő az adott oszlop két legkisebb elemének abszolút különbségével, azaz ha t'_j és t''_j jelöli a j-edik oszlop két legkisebb elemét, akkor a

$$t_j \;=\; |\, t'_j - t''_j \,|$$

A kiszámolt $t_j,\ j = 1, 2, \ldots, n,$ bütetésekből ki kell választani a legnagyobbat, majd a hozzá tartozó oszlopban ki kell választani a legkisebb költségű cellát. Az ilyen módon meghatározott cellába be kell vezetni a lehetőleg legnagyobb szállítást. Itt lesz az első bázisváltozó. Majd ugyanúgy, mint az előző módszerek esetén, változtassuk a megfelelő készletet, keresletet és húzzuk át a megfelelő oszlopot vagy sort. Az adott eljárást addig kell ismételnünk, amíg a táblázatban nem marad egyetlen egy át nem húzott sor vagy oszlop sem.

Illusztráljuk a módszer működését a $D = ||d_{ij}||_{4\times 4}$ mátrixot, az $a = (a_1, a_2, a_3, a_4)$ keresletet és a $b = (b_1, b_2, b_3, b_4)^T$ készletet tartalmazó a(z) 9. táblázatban feltüntetett numerikus példa alapján.

Elsőször minden oszlophoz rendeljünk hozzá büntetéseket (lásd. a(z) 15. táblázat). Például az első oszlopban a két legkisebb elem $d_{21} = 3$ és $d_{41} = 4$, ezért $t_1 = 1$. Így,

$$t_1 = 1,\ \ t_2 = 1,\ \ t_3 = 2,\ \ t_4 = 2.$$

Majd válasszunk legnagyobb büntetést, pl. t_4, és a 4.-edik oszlopban válasszuk a legkisebb d_{i4} elemet, azaz $d_{14} = 1$. Ezután az ilyen módon meghatározott $(1, 4)$ cellába vezessük be a lehető legnagyobb szállítást, azaz

$$x_{14} \;=\; \min\{b_1, a_4\} \;=\; \min\{200, 50\} \;=\; 50.$$

Végül változtassuk meg a megfelelő módon az a_4 keresletet és b_1 készletet:

$$b_1 \;\to\; b_1 - a_4 \;=\; 200 - 50 \;=\; 150$$
$$a_4 \;\to\; 0$$

Az eredményeket írjuk be a(z) 16. táblázatba. Az adott eljárást ismételve sorban kapjuk a(z) 17, 18, 19 és végül a(z) 20 táblázatokat.

	1	2	3	4	
1	8	6	6	1	200
2	3	4	6	8	100
3	7	3	8	9	100
4	4	12	4	3	100
	120	150	180	50	
	$4-3=1$	$4-3=1$	$6-4=2$	$3-1=2$	

15. Táblázat. Vogel-módszer – 1. tábla

	1	2	3	4	
1	8	6	6	**50** 1	150
2	3	4	6	8	100
3	7	3	8	9	100
4	4	12	4	3	100
	120	150	180	×	
	$4-3=1$	$4-3=1$	$6-4=2$	–	

16. Táblázat. Vogel-módszer – 2. tábla

A végső tábla megtekinthető a 20. táblázatban. A kapott lehetséges

	1	2	3	4	
1	8	6	6	50 1	150
2	3	4	6	8	100
3	7	3	8	9	100
4	4	12	**100** 4	3	×
	120	150	80	×	
	$7-3=4$	$4-3=1$	$8-6=2$	−	

17. Táblázat. Vogel-módszer – 3. tábla.

	1	2	3	4	
1	8	6	6	**50** 1	150
2	**100** 3	4	6	8	×
3	7	3	8	9	100
4	4	12	**100** 4	3	×
	20	150	80	×	
	$8-7=1$	$6-3=3$	$8-6=2$	−	

18. Táblázat. Vogel-módszer – 4. tábla.

bázismegoldás: $x_{11}=20$, $x_{12}=50$, $x_{13}=80$, $x_{14}=50$, $x_{21}=100$, $x_{32}=100$, $x_{43}=100$.

Összefoglalás: A most megtárgyalt három módszer közül a lehetséges bázismegoldás előállítására az északnyugati sarok módszer igényli a legkevesebb, és a Vogel-módszer a legtöbb erőfeszítést. Alapos kutatások megmutatták, hogy amikor a Vogel-módszerrel állítjuk elő az induló lehetséges

	1	2	3	4	
1	8	6	6	**50** 1	150
2	**100** 3	4	6	8	×
3	7	**100** 3	8	9	×
4	4	12	**100** 4	3	×
	20	50	80	×	
	8	6	6	–	

19. Táblázat. Vogel-módszer – 5. tábla.

	1	2	3	4	
1	20 8	50 6	80 6	50 1	×
2	100 3	4	6	8	×
3	7	100 3	8	9	×
4	4	12	100 4	3	×
	×	×	×	×	

20. Táblázat. Vogel-módszer – végső tábla.

bázismegoldást, akkor lényegesen kevesebb további iterációs lépésre van szükség, mint a másik két módszer alkalmazásakor. Ezért nagyméretű szállítási feladatok esetén az első lehetséges bázismegoldás előállítására az északnyugati sarok módszert és a minimális költség (vagy maximális profit) módszert csak ritkán használják.

4. Numerikus példa

Tekintsük a következő 3×4 méretű numerikus példát:

$$Q(x') = \frac{P(x)}{D(x)} = \frac{\sum_{i=1}^{3}\sum_{j=1}^{4} p_{ij}x_{ij} + p_0}{\sum_{i=1}^{3}\sum_{j=1}^{4} d_{ij}x_{ij} + d_0} \longrightarrow \max \tag{147}$$

a következő feltételek mellett

$$\left.\begin{array}{rcl} x_{11}+x_{12}+x_{13}+x_{14} & \leq & 150, \\ x_{21}+x_{22}+x_{23}+x_{24} & \leq & 250, \\ x_{31}+x_{32}+x_{33}+x_{34} & \leq & 200, \end{array}\right\} \tag{148}$$

$$\left.\begin{array}{rcl} x_{11}+x_{21}+x_{31} & \geq & 150, \\ x_{12}+x_{22}+x_{32} & \geq & 250, \\ x_{13}+x_{23}+x_{33} & \geq & 50, \\ x_{14}+x_{24}+x_{34} & \geq & 150; \end{array}\right\} \tag{149}$$

$$x_{ij} \geq 0, \quad i = 1,2,3, \quad j = 1,2,3,4, \tag{150}$$

ahol $p_0 = 100$, $d_0 = 120$, meg p_{ij} és d_{ij} együtthatók a következők:

p_{ij}	1	2	3	4
1	10	14	8	12
2	8	12	14	8
3	9	6	15	9

d_{ij}	1	2	3	4
1	15	12	16	8
2	10	6	13	12
3	13	15	12	10

Vegyünk figyelembe, hogy az adott feladat egyensúlyozott, és ezért semmilyen fiktív változókra nincs szükség.

Az induló lehetséges bázismegoldás meghatározásához használt maximális profit módszert a(z) 21. táblázatban feltüntetett megoldáshoz vezet. A maximális profit módszer alkalmazása során kaptuk a következő

$$x_{12} = 150, \quad x_{22} = 100, \quad x_{24} = 150, \quad x_{31} = 150, \quad x_{33} = 50$$

5 darab bázisváltozót. Mivel definició szerint egy 3×4 méretű szállítási feladat bázismegoldásában szerepel $3+4-1 = 6$ bázisváltozó, ezért a kapott lehetséges megoldás nem bázismegoldása a megoldandó feladatnak. Ilyenkor a bázisba be kell vezetni még egy változót, pl.

$$x_{11} = 0.$$

Az ilyen módon összeállított lehetséges bázismegoldás degenerált, és tartalmazza a következő bázis céllákat:

$$J_B = \{(1,1),\ (1,2),\ (2,2),\ (2,4),\ (3,1),\ (3,3)\}.$$

	1	2	3	4	
1	10 **0** 15	14 **150** 12	8 16	12 8	150
2	8 10	12 **100** 6	14 13	8 **150** 12	250
3	9 **150** 13	6 15	15 **50** 12	9 10	200
	150	250	50	150	

21. Táblázat. Hurokszerkesztéses szimplex módszer – Induló lehetséges bázismegoldás.

Ennek a lehetséges bázismegoldásnak megfelelnek a következő cél-függvény értékek:

$$P(x) = 6700, \quad D(x) = 6870, \quad Q(x) = 6700/6870 \ (\approx 0.975255) \ .$$

Most a(z) (141)-(142) képletek használatával állítsuk össze a következő egyeletrendszereket:

$$\left.\begin{array}{rcl} u'_1 + v'_1 &=& 10, \\ u'_1 + v'_2 &=& 14, \\ u'_2 + v'_2 &=& 12, \\ u'_2 + v'_4 &=& 8, \\ u'_3 + v'_1 &=& 9, \\ u'_3 + v'_3 &=& 15, \end{array}\right\} \tag{151}$$

$$\left.\begin{array}{rcl} u''_1 + v''_1 &=& 15, \\ u''_1 + v''_2 &=& 12, \\ u''_2 + v''_2 &=& 6, \\ u''_2 + v''_4 &=& 12, \\ u''_3 + v''_1 &=& 13, \\ u''_3 + v''_3 &=& 12, \end{array}\right\} \tag{152}$$

A(z) (151) és (152) rendszerben legyen $u'_1 = 0$ és $u''_1 = 0$, akkor kapjuk a következőt:

$$u'_1 = 0, \ u'_2 = -2, \ u'_3 = -1, \quad v'_1 = 10, \ v'_2 = 14, \ v'_3 = 16, \ v'_4 = 10,$$

és

$$u''_1 = 0, \ u''_2 = -6, \ u''_3 = -2, \quad v''_1 = 15, \ v''_2 = 12, \ v''_3 = 14, \ v''_4 = 18.$$

Ezeket a változókat használjuk fel a Δ'_{ij} és Δ''_{ij} (lásd (143)) értékek meghatározásához a következő nem-bázis cellákban:

$$J_N = \{(1,3), (1,4), (2,1), (2,3), (4,2), (4,4)\}.$$

Kapjuk, hogy

$$\begin{array}{rclcrcr}
\Delta'_{13} & = & u'_1 + v'_3 - p_{13} & = & 0+16-8 & = & 8, \\
\Delta'_{14} & = & u'_1 + v'_4 - p_{14} & = & 0+10-12 & = & -2, \\
\Delta'_{21} & = & u'_2 + v'_1 - p_{21} & = & -2+10-8 & = & 0, \\
\Delta'_{23} & = & u'_2 + v'_3 - p_{23} & = & -2+16-14 & = & 0, \\
\Delta'_{32} & = & u'_3 + v'_2 - p_{32} & = & -1+14-6 & = & 7, \\
\Delta'_{34} & = & u'_3 + v'_4 - p_{34} & = & -1+10-9 & = & 0,
\end{array}$$

$$\begin{array}{rclcrcr}
\Delta''_{13} & = & u''_1 + v''_3 - d_{13} & = & 0+14-16 & = & -2, \\
\Delta''_{14} & = & u''_1 + v''_4 - d_{14} & = & 0+18-8 & = & 10, \\
\Delta''_{21} & = & u''_2 + v''_1 - d_{21} & = & -6+15-10 & = & -1, \\
\Delta''_{23} & = & u''_2 + v''_3 - d_{23} & = & -6+14-13 & = & -5, \\
\Delta''_{32} & = & u''_3 + v''_2 - d_{32} & = & -2+12-15 & = & -5, \\
\Delta''_{34} & = & u''_3 + v''_4 - d_{34} & = & -2+18-10 & = & 6.
\end{array}$$

Továbbá a kapott Δ'_{ij}, Δ''_{ij} és (144) képlet használatával meghatározhatjuk a $\Delta_{ij}(x)$ értékeket:

$$\begin{array}{rclcr}
\Delta_{13}(x) & = & \Delta'_{13} - Q(x)\,\Delta''_{13} & = & 9\dfrac{653}{687}, \\
\Delta_{14}(x) & = & \Delta'_{14} - Q(x)\,\Delta''_{14} & = & -11\dfrac{517}{687}, \\
\Delta_{21}(x) & = & \Delta'_{21} - Q(x)\,\Delta''_{21} & = & \dfrac{670}{687}, \\
\Delta_{23}(x) & = & \Delta'_{23} - Q(x)\,\Delta''_{23} & = & 4\dfrac{602}{687}, \\
\Delta_{32}(x) & = & \Delta'_{32} - Q(x)\,\Delta''_{32} & = & 11\dfrac{602}{687}, \\
\Delta_{34}(x) & = & \Delta'_{34} - Q(x)\,\Delta''_{34} & = & -5\dfrac{195}{229}.
\end{array}$$

Mivel a kapott nem-bázis indexű $\Delta_{ij}(x)$ értékek között vannak negatív értékűek, ezért az aktuális lehetséges bázismegoldás nem optimális megoldása a megoldandó feladatnak. Ezért a negatív nem-bázis indexű $\Delta_{ij}(x)$ értékekből ki kell választani az egyiket, és hozzátartozó nem-bázis x_{ij} változót be kell vezetni a bázisba.

Válasszuk az x_{14} változót. Majd az $(1,4)$ indexű cellába vezessünk be θ értékű szállítást, és ebből a cellából kiindulva állítsunk össze egy hurkot. Ezeknek a műveleteknek az eredményét megtekinthetjük a(z) 22. táblázátban. Az összaállított hurok segítségével meghatározhatjuk a θ értékét:

$$\theta = \min\{x_{12},\ x_{24}\} = \min\{150,\ 150\} = 150.$$

Jegyezzük meg, hogy a fenti képletben a θ érték élérhető két külön $(1,2)$ és $(2,4)$ cellában. Ez azt jelenti, hogy az aktuális bázisban szereplő x_{12} és x_{24} változóból nekünk kell kiválsztani egyet, és azt kivezetni a bázisból, a másik pedig degenerált változóként marad a bázisban nulla értékkel.

	1	2	3	4	
1	10 0 15	14 $150-\theta$ 12 $\downarrow$	8 16	12 $\leftarrow$ θ 8	150
2	8 10	12 $100+\theta$ $\rightarrow$ 6	14 13	8 $\uparrow$ $150-\theta$ 12	250
3	9 150 13	6 15	15 50 12	9 10	200
	150	250	50	150	

22. Táblázat. Hurokszerkesztéses szimplex módszer – 1. tábla

Válasszuk az x_{24} változót, és vezessük azt ki a bázisból. Így az új bázis tartalmazza a

$$J_B = \{(1,1),(1,2),(1,4),(2,2),(3,1),(3,3)\},$$

cellákat, a nem-bázis callák pedig a következők:

$$J_N = \{(1,3),(2,1),(2,3),(2,4),(3,2),(3,4)\}.$$

A szimplex iteráció végrehajtása után kapjuk a a(z) 23. táblázatban feltüntetett táblát. Az új lehetséges bázismegoldás:

	1	2	3	4	
1	10 0 15	14 0 12	8 16	12 150 8	150
2	8 10	12 250 6	14 13	8 12	250
3	9 150 13	6 15	15 50 12	9 10	200
	150	250	50	150	

23. Táblázat. Hurokszerkesztéses szimplex módszer – 2. tábla

$$x_{11}=0,\ x_{12}=0,\ x_{14}=150,\ x_{22}=250,\ x_{31}=150,\ x_{33}=50$$

és a hozzátartozó cél-függvény értékek:

$$P(x)=7000,\ \ D(x)=5370,\ \ Q(x)=7000/5370\ (\approx 1.303538).$$

A továbbiakban az új bázisban állítsuk össze a következő egyenletrendszereket:

$$\begin{array}{rcl} u_1' + v_1' &=& 10, \\ u_1' + v_2' &=& 14, \\ u_1' + v_4' &=& 12, \\ u_2' + v_2' &=& 12, \\ u_3' + v_1' &=& 9, \\ u_3' + v_3' &=& 15, \end{array}$$

és

$$\begin{array}{rcl} u_1'' + v_1'' &=& 15, \\ u_1'' + v_2'' &=& 12, \\ u_1'' + v_4'' &=& 8, \\ u_2'' + v_2'' &=& 6, \\ u_3'' + v_1'' &=& 13, \\ u_3'' + v_3'' &=& 12. \end{array}$$

Oldjuk meg ezeket az egyenletrendszereket, és kapjuk a következő megoldásokat:

$$u_1' = 0,\ u_2' = -2,\ u_3' = -1, \quad v_1' = 10,\ v_2' = 14,\ v_3' = 16,\ v_4' = 12,$$

és

$$u_1'' = 0,\ u_2'' = -6,\ u_3'' = -2, \quad v_1'' = 15,\ v_2'' = 12,\ v_3'' = 14,\ v_4'' = 8,$$

amelyek alapján kiszámolhatjuk a

$$\begin{array}{rcrcrcl} \Delta_{13}' &=& u_1' + v_3' - p_{13} &=& 0 + 16 - 8 &=& 8, \\ \Delta_{21}' &=& u_2' + v_1' - p_{21} &=& -2 + 10 - 8 &=& 0, \\ \Delta_{23}' &=& u_2' + v_3' - p_{23} &=& -2 + 16 - 14 &=& 0, \\ \Delta_{24}' &=& u_2' + v_4' - p_{24} &=& -2 + 12 - 8 &=& 2, \\ \Delta_{32}' &=& u_3' + v_2' - p_{32} &=& -1 + 14 - 6 &=& 7, \\ \Delta_{34}' &=& u_3' + v_4' - p_{34} &=& -1 + 12 - 9 &=& 2, \end{array}$$

értékeket, majd a

$$\begin{array}{rcrcrcl} \Delta_{13}'' &=& u_1'' + v_3'' - d_{13} &=& 0 + 14 - 16 &=& -2, \\ \Delta_{21}'' &=& u_2'' + v_1'' - d_{21} &=& -6 + 15 - 10 &=& -1, \\ \Delta_{23}'' &=& u_2'' + v_3'' - d_{23} &=& -6 + 14 - 13 &=& -5, \\ \Delta_{24}'' &=& u_2'' + v_4'' - d_{24} &=& -6 + 8 - 12 &=& -10, \\ \Delta_{32}'' &=& u_3'' + v_2'' - d_{32} &=& -2 + 12 - 15 &=& -5, \\ \Delta_{34}'' &=& u_3'' + v_4'' - d_{34} &=& -2 + 8 - 10 &=& -4, \end{array}$$

értékeket, és végül a

$$\begin{aligned}
\Delta_{13}(x) &= \Delta'_{13} - Q(x)\,\Delta''_{13} = 10\frac{326}{537},\\
\Delta_{21}(x) &= \Delta'_{21} - Q(x)\,\Delta''_{21} = 1\frac{163}{537},\\
\Delta_{23}(x) &= \Delta'_{23} - Q(x)\,\Delta''_{23} = 6\frac{278}{537},\\
\Delta_{24}(x) &= \Delta'_{24} - Q(x)\,\Delta''_{24} = 15\frac{19}{537},\\
\Delta_{32}(x) &= \Delta'_{32} - Q(x)\,\Delta''_{32} = 13\frac{278}{537},\\
\Delta_{34}(x) &= \Delta'_{34} - Q(x)\,\Delta''_{34} = 7\frac{115}{537}.
\end{aligned}$$

értékeket.

Mivel az összes nem-bázis indexű $\Delta_{ij}(x)$ érték nem negatív, azaz

$$\Delta_{ij}(x) \geq 0,\ (ij) \in J_N$$

ez azt jelenti, hogy az aktuális lehetséges bázismegoldás optimális.

VI. Fejezet
A WinGULF programcsomag

WinGULF - is a **G**eneral, **U**ser-friendly **L**inear and linear-**F**ractional programming package for **Win**dows.

Eredetileg a WinGULF programcsomag az ausztrál származású GULP elnevezésű lineáris programozási programcsomag leszármazottja. A GULP programcsomagot David J. Pannell[1] fejlesztette ki MS-DOS-ban az 1990-es évek elején Turbo Pascal nyelven. Világhíres és népszerű LP eszköz volt a felsőoktatási intézményekben [**184**] és mindenütt, ahol szerkeszteni és megoldani kellett lineáris programozási feladatokat folytonos változókkal.

A jelen jegyzet szerzH oje 1993-ban a David J. Pannell együttműködésével továbbfejlesztette a GULP-ot úgy, hogy "GULF" elnevezést kapott az új programcsomag, amely már képes megoldani hiperbolikus programozási feladatot is. A GULF programcsomag első verziója sikeresen átment az "European Journal of Operational Research" című szakfolyóirat operációkutatási szoftver szerkesztőségében az alkalmazhatósági teszteléseken, és ott pozitív véleményt kapott [**185**]. Később a Borland Delphi fejlesztői programcsomag megjelenésével a GULF "át lett ültetve" az MS-Windows alá. Az 1998 óta WinGULF programcsomag rendelkezik saját "branch-and-bound" motorral, ami lehetővé teszi az egészértékű változókat tartalmazó hiperbolikus programozási feladatok megoldását is.

Manapság a WinGULF programcsomag speciális "Student Edition" elnevezésű verziója szabadon letölthető a szerző

http:\\www.inf.unideb.hu\ ∼bajalinov\

című Web-lapjáról.

A jelen fejezetben a szóbanforgó programcsomagot ismertetjük meg.

[1]The University of Western Australia, School of Agriculture, Nedlands W.A. 6009 Australia

1. A Programcsomag rövid áttekintése

A WinGULF programcsomag teljes elnevezése a "**G**eneral **U**ser-friendly **L**inear and linear-**F**ractional programpackage for **Win**dows".

Más operációkutatási szoftverekkel szemben a WinGULF legfontosabb különlegessége az, hogy a programcsomag képes megoldani nem csak lineáris programozási feladatokat, hanem hiperbolikus feladatokat is. A WinGULF hagyományos menürendszerrel rendelkezik, amely segítségével a felhasználó elérheti az összes, a csomagba beépített eszközt. A táblázatkezelő szerű felületen megjelenített feladatot lehet szerkeszteni, menteni, beolvasni. A viszonylag egyszerű aritmetikai számolások végrehajtásához használhatjuk a beépített számológépet. Az adatok begépelése/szerkesztése során azok helyességi vizsgálata "on-fly" módban történik. Adatokat lehet begépelni vagy szerkeszteni manuálisan, illetve beolvasni lemezről vagy menteni lemezre az MPS[2] formátumú állományba (állományból).

A csomag rendelkezik beépített 2-fázisú "primál szimplex" eljárással és "korlátozás-és-szétválasztás" motorral. Mindez lehetővé teszi lineáris és hiperbolikus programozási feladatok megoldását nem csak folytonos változókkal, hanem egészértékű változók esetén is.

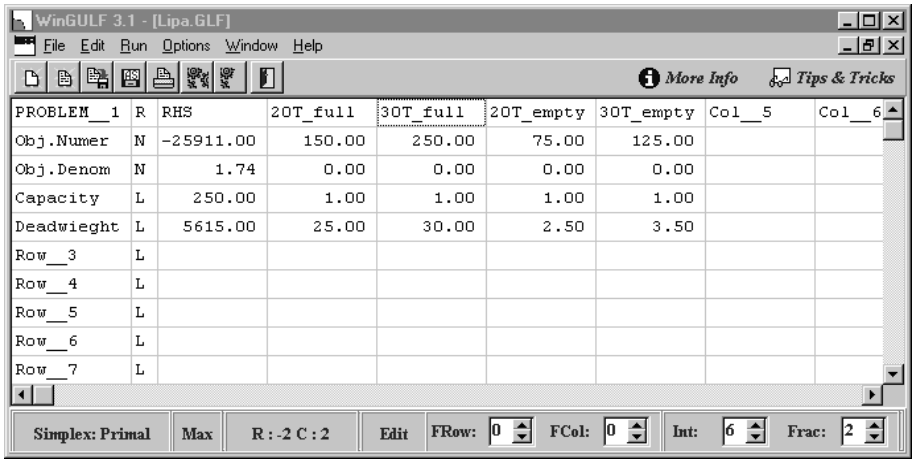
A szimplex módszer két módban futtatható: az egyik az ún. **automatic**, a másik pedig a **step-by-step**. Az első módban a szimplex eljárás végrehajtása végig automatikusan történik. A feladattól függően vagy megkapjuk az optimális megoldást, vagy megfelelő arról szóló üzenetet, hogy a feladat valamely ok miatt nem megoldható. A másik módban történő futtatás esetén az eljárás végrehajtása lépésenként történik. Minden iteráció után a csomag megáll és várakozik, ilyenkor a felhasználó kiválaszthatja a következő generáló oszlopot.

A maximális méretű feladat, amelyet képes megoldani az Intenetről szabadon letölthető verzió, 50 feltételt és 50 változót tartalmaz, ebből az 50 változóból maximum 25 változó lehet egészértékű.

A csomag telepítéséhez $1,5Mb$ szabad terület elegendő.

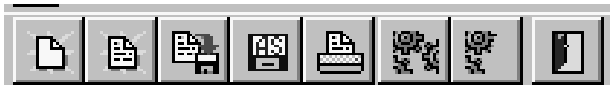
Amennyiben a megoldandó folytonos feladat be van gépelve vagy olvasva (lásd. 1. ábra), máris a *Run* gombon történő kattintással (lásd. 2. ábra) vagy *Run*→*Run* menüpont kiválasztásával (forró gomb-*F8*) lehet indítani a szimplex módszert. Ilyenkor a program a szimplex eljárást "*automatic*" módban indítja, és így hajtja végre a módszert. A szimplex módszer végrehajtása

[2]MPS – a "Mathematical Programming System" rövidetése. Ez egy speciális szabványosított Operációkutatási formátum, amely szerinti szöveges állományokban szokták tárolni a matematikai programozási feladatokat.



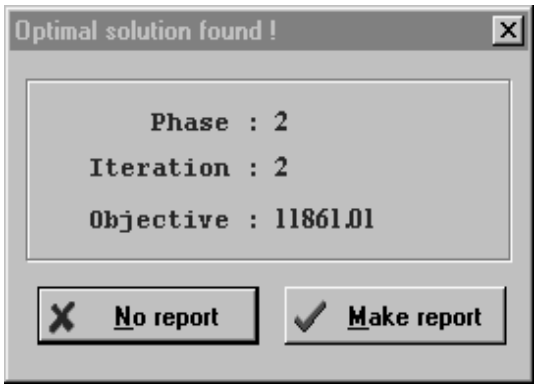
PROBLEM__1	R	RHS	20T_full	30T_full	20T_empty	30T_empty	Col__5	Col__6
Obj.Numer	N	-25911.00	150.00	250.00	75.00	125.00		
Obj.Denom	N	1.74	0.00	0.00	0.00	0.00		
Capacity	L	250.00	1.00	1.00	1.00	1.00		
Deadwieght	L	5615.00	25.00	30.00	2.50	3.50		
Row__3	L							
Row__4	L							
Row__5	L							
Row__6	L							
Row__7	L							

1. ábra. WinGULF – Folytonos LFP feladat



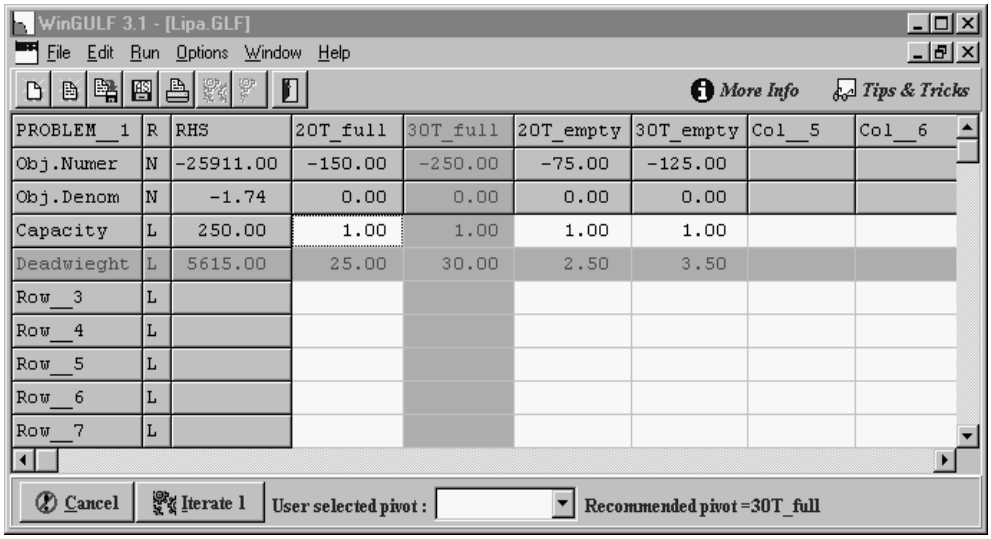
2. ábra. WinGULF – Fő funkcionális gombok.

során a képernyőn megjelenített párbeszédablakban láthatjuk a statisztikát: a fázis neve, iterációs lépés sorszáma és a cél-függvény aktuális értéke (lásd. 3. ábra). Abban az esetben, ha szeretnénk megtekinteni a szimplex módszer



3. ábra. WinGULF – Status window.

futási folyamatát, akkor a *Run by Step* gombot kell választanunk (menüpont: *Run→Run by Step*, forró gomb: *F9*). Ilyenkor a képernyőn megjelenik az induló szimplex tábla (lásd. 4. ábra), majd a program megáll és várakozik. A *Run by Step* mód kiválasztása esetén az ablak alsó részén a következő vezérlési eszközöket vehetjük igénybe (lásd. 4. ábra):
Cancel gomb (megszakítja a szimplex eljárást, és átadja a vezérlést a fő



4. ábra. WinGULF – Step-by-Step mode.

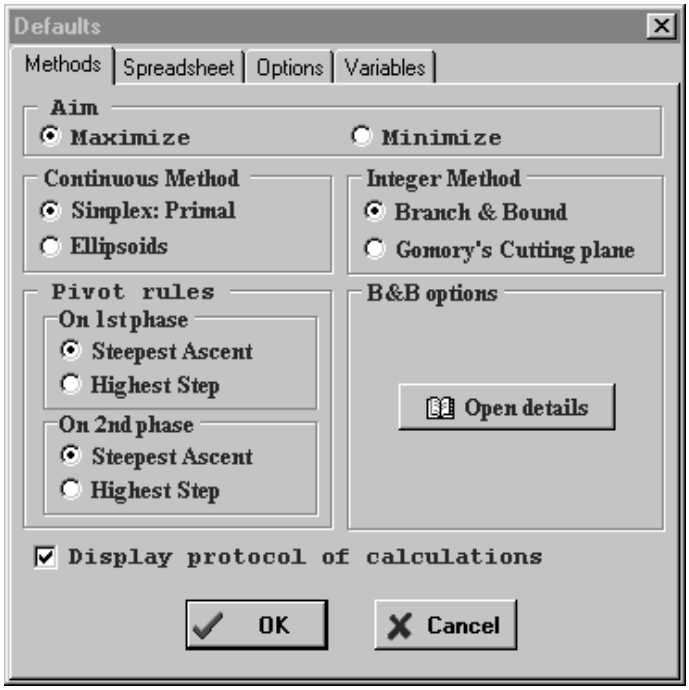
ablakba),
Iterate $< K >$ gomb (végrehajtja a szimpex módszer $< K >$.-edik iterációját),
User selected pivot lefelé nyíló lista combo box (a generáló oszlop kiválasztásához).

Ezenkívül itt találhatjuk a program által javasolt generáló oszlop azonosítóját is. A program által megjelölt generáló oszlop kiválasztása az "*Options*" (menü: *Options*→*Defaults*, majd *Methods* lap) ablakban rögzített szabály szerint történik.

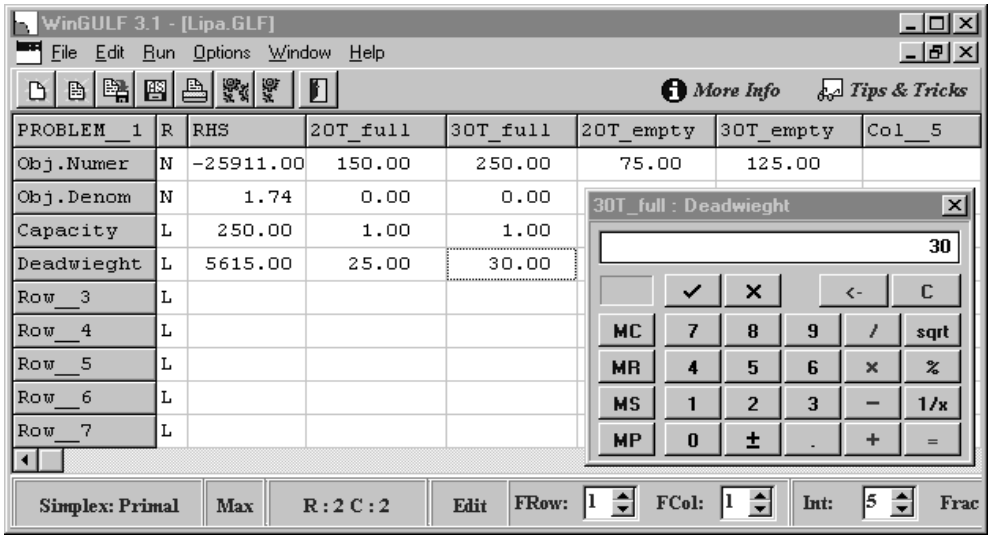
A *Defaults* párbeszéd ablak *Methods* nevű lapja megtekinthető a(z) 5. ábrán.

2. Szerkesztő

A WinGULF programcsomag középpontja a táblázatkezelő típusú felülettel rendelkező szerkesztő ablak, amelyben történik az új és meglévő feladatok szerkesztése. A feladathoz tartozó együtthatóknak a megfelelő cellákba történő bevétele két módon történhet: az egyik "manuális" – azaz numerikus vagy szöveges adatot úgy kell begépelni közvetlenül a cellába, ahogy van; a másik mód pedig "számológépes". Az utóbbi módot csak numerikus celláknál vehetjük igénybe az egér jobb gombjának használatával (6. ábra). A beépített számológép ablakának felső részén láthajuk az aktuális cella pozíciója a képernyőn.



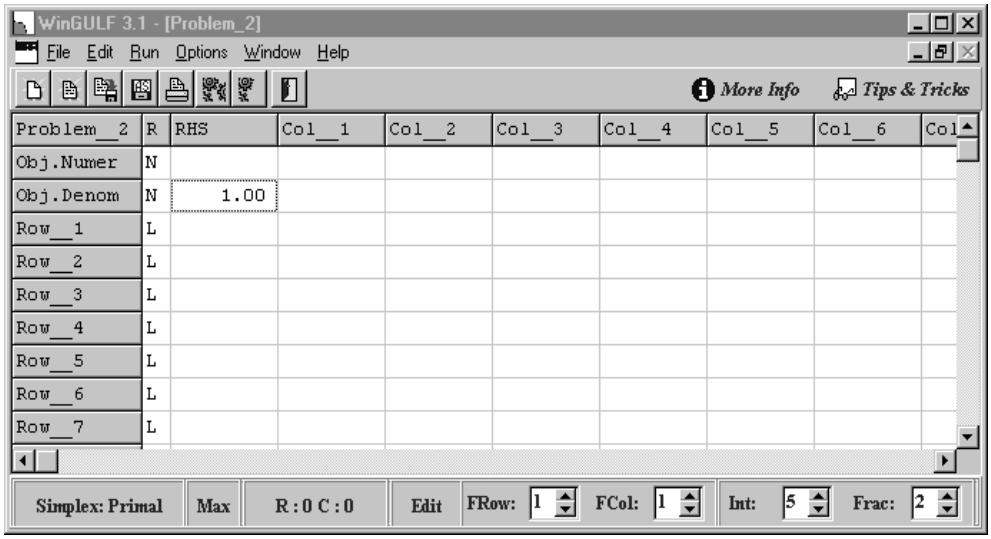
5. ábra. WinGULF – Defaults→Methods lap.



6. ábra. WinGULF – Beépített számológép.

A szerkesztési szakaszon a képernyő tartalmazza az aktuális feladatot vagy üres rácsos felületet új feladat esetén (lásd. 7. ábra).

A rácsos felület felső bal oldali cellája tartalmazza a feladat azonosítóját. Ez a cella (a 7. ábrán "Problem2") csak akkor érhető el, ha az ablak alsó



7. ábra. WinGULF – Új feladat.

részén található és "FRow" ("Fixed Rows") meg "FCol" ("Fixed columns") cimkékkel ellátott vezérlők nullára vannak beállítva (lásd. 8. ábra).

8. ábra. WinGULF – Rögzített sorok és oszlopok beállítására szolgáló vezérlők.

Az "RHS" oszlop és "Obj.Denom" sor kereszteződésében álló "1.00" szám az alapértelmezett értéke a d_0 együtthatónak. Ennek a sornak a többi együtthatói csupa 0.00 értékűek, de mivel "0.00" értékű együtthatókat a szerkesztő rács nem jeleníti meg, ezért a megfelelő cellák üresek. Tehát az új feladat esetén

$$D(x) = 0.00\,x_1 + 0.00\,x_2 + \ldots + 0.00\,x_n + 1.00$$

Mivel ilyenkor $D(x) = 1$, ez azt jelenti, hogy az alapértelmezés szerint az új feladat lineáris és a lineáris programozási feladat cél-függvényének együtthatói a "Obj.Numer" sorban vannak. Ez addig marad így, amig "Obj.Denom" sorban az összes $d_j,\ j = 1, 2, \ldots, n$ egyenlő nullával. Egyébként a cél-függvény tört alakú és a feladat hiperbolikus.

A szerkesztésnél ügyelnünk kell arra, hogy minden együttható saját cellába kerüljön a következőknek megfelelően:

	Pozició	
Együttható	Sor	Oszlop
p_0	$Obj.Numer$	RHS
p_1	$Obj.Numer$	$Col\ 1$
p_2	$Obj.Numer$	$Col\ 2$
$\vdots$	$\vdots$	$\vdots$
p_n	$Obj.Numer$	$Col\ n$
d_0	$Obj.Denom$	RHS
d_1,	$Obj.Denom$	$Col\ 1$
$\vdots$	$\vdots$	$\vdots$
d_n	$Obj.Denom$	$Col\ n$
b_1	$Row\ i$	RHS
b_2	$Row\ 2$	RHS
$\vdots$	$\vdots$	$\vdots$
b_m	$Row\ m$	RHS
a_{ij}	$Row\ i$	$Col\ j$

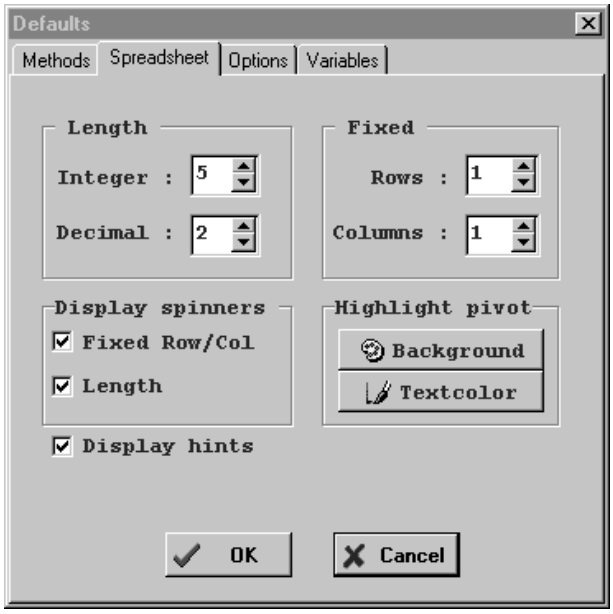
A bal oldali szélső oszlopban és a legfelső sorban található sorcimkék, illetve oszlopcimkék (”Row 1”, ”Row 2”, ”Col 1”, ”Col 2”, stb.) szerkesztéséhez a rögzített oszlopok és rögzített sorok beállítására szolgáló vezérlőket át kell állítanunk nullára (lásd. 8. ábra).

Természetesen a Windows hagyományoknak megfelelően a feladatot tartalmazó ablak vizszintes és függőleges görgetősávokkal rendelkezik arra az esetre, ha a feladat mérete a képernyőhez képest túl nagy.

A WinGULF legalsó részében találhatunk még két darab hasznos vezérlőt (lásd. 9. ábra), amelyek segítségével beállíthatjuk a numerikus értékek megjelenítési pontosságát.

9. ábra. WinGULF – Numerikus érétekek megjelenítési pontosságát beállító vezérlők.

A munkafelület testreszabását végezhetjük az *Options* párbeszéd ablak *Spreadsheet* oldalán (menü: *Options→Defaults→Spreadsheet*) (lásd. 10. ábra).

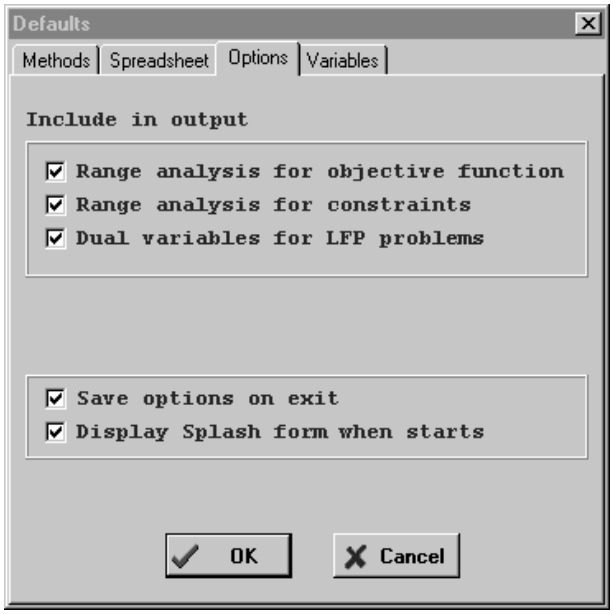


10. ábra. WinGULF – Options ablak.

Ha a feladatot sikerült megoldani, és a megjelent párbeszéd ablakban a *Make Report* gombra kattintottunk (lásd. 3. ábra), akkor a WinGULF legenerálja a kapott optimális megoldásról szóló jelentést (lásd. 3. és 4. alfejezet). Ennek a jelentésnek a tartalma függ attól, hogy a feladat tartalmaze csak folytonos változókat, vagy vannak benne egészértékű változók is. Ezenkívül a jelentés tartalma konfigurálható az "*Options*" ablakban is (lásd. 11. ábra).

3. Folytonos feladatok

A jegyzet jelen részében olyan LP és HP feladatok WinGULF-ban történő kezelésével fogunk megismerkedni, amelyek csak folytonos változókat tartalmaznak.



11. ábra. WinGULF – Opciók.

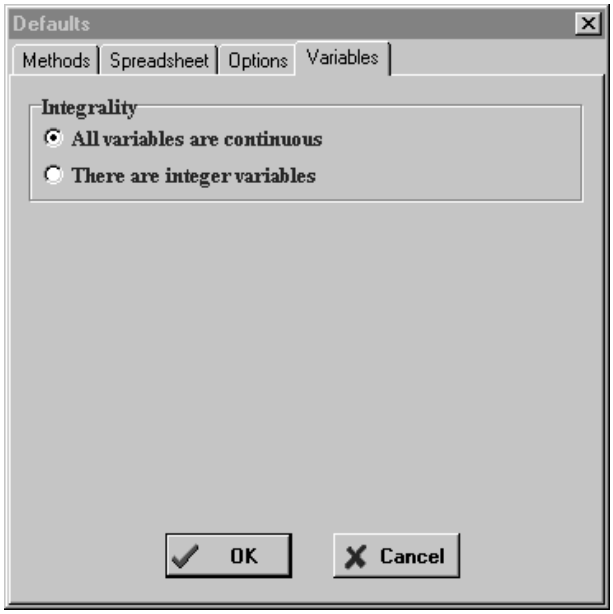
3.1. Bevétel és fő Opciók

Ahogy már említettük az előző alfejezetben, a WinGULF-ban szerkeszteni (kezelni) lehet meglévő feladatokat és új feladatokat. A választástól függően ehhez használhatjuk a megfelelő gombokat (2. ábra) vagy a megfelelő menüpontokat (*File→New* vagy *File→Open*).

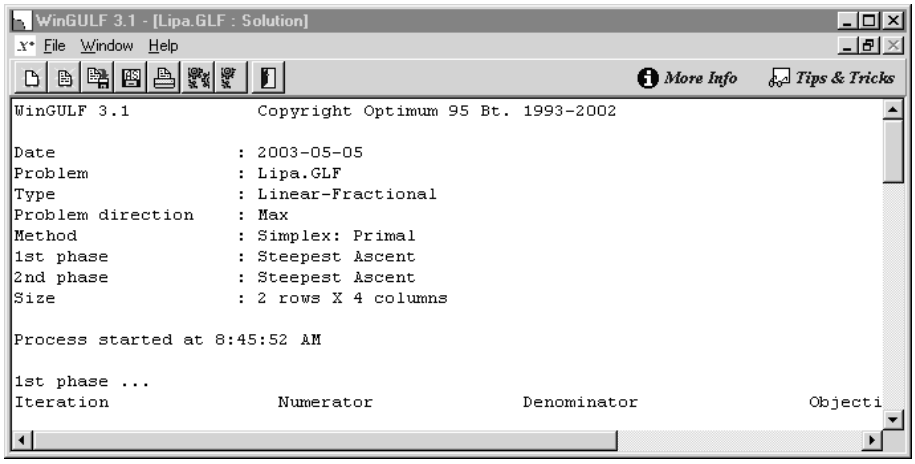
A feladat folytonosságának beállításához használhatjuk az *"All variables are continuous"* opciót a *"Defaults"* párbeszéd ablakban található *"Variables"* című lapon (lásd. 12. ábra). A feladat szerkesztésének befejeződése után a feladatot menthetjük MPS-formátumú szöveges állományba és/vagy kinyomtathatjuk a nyomtatón.

3.2. Kimenet

Ha a feladatot sikerült megoldani, és a megjelenített *"Status window"* ablakban kiválasztottuk a *"Make Report"* gombot, akkor a WinGULF legenerálja a kapott megoldásról szóló jelentést, majd megjeleníti azt egy külön szöveges ablakban (lásd. 13. ábra). A jelentés szerkezete szabványos MPS formátumú, így a hordozhatóságának köszönhetően a jelentés nemcsak más optimálizálási eszközekkel olvasható és kezelhető, hanem más rendszerekben (UNIX, LINUX, SOLARIS, stb.) is.



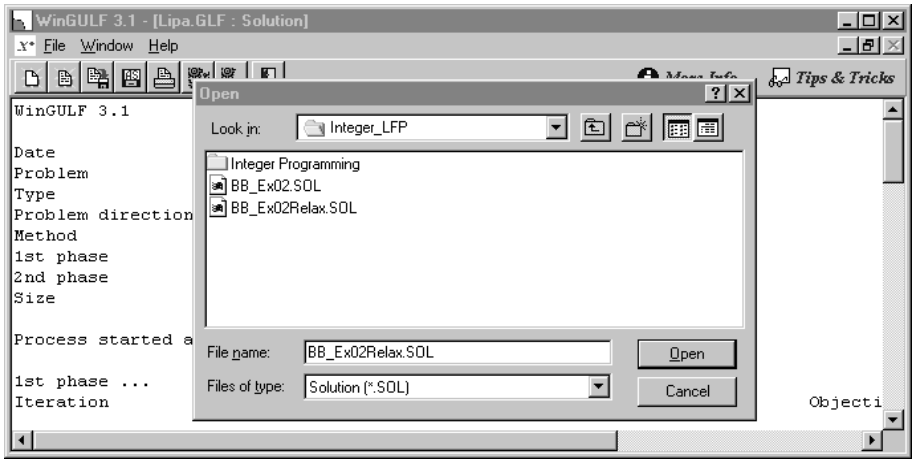
12. ábra. WinGULF – "Variables" lap.



13. ábra. WinGULF – Folytonos feladatat jelentése.

A megjelenített jelentést kinyomtathatjuk vagy menthetjük szöveges állományba (".*SOL*" alapértelmezési kiterjesztéssel). Ha szükségünk van egy korábban mentett jelentésre, akkor a "File" menüpont segítségével nyithatjuk meg a megfelelő párbeszéd ablakot (lásd.14. ábra) és azzal kiválaszthatjuk a kívánt jelentést.

A WinGULF által összeállított jelentés két nagyobb részből áll:



14. ábra. WinGULF – Jelentés nyítása.

(1) a jelentés felső része általános adatokat és statisztikát tartalmaz – itt találhatjuk az aktuális dátumot, a feladat nevét, az optimálizálási célt (Max/Min), igénybe vett módszer(eke)t, a feladat méretét, és a folyamat protokollját (az utóbbi kikapcsolható), a végrehajtott iterációs lépések számát stb.,
(2) a jelentés második része a kapott optimális megoldásról szóló részletes információt foglal magában – itt találhatjuk a cél-függvény optimális értéket, primál és duális változókot, Δ_j értékeket, érzékenységi vizsgálat eredményeit stb.

A lineáris programozási feladat esetén a jelentés második része négy szakaszból áll: az első szakasz oszlopokhoz tartozó információt szolgál, a második pedig a sorokról szól. Utána kövekezik az érzékenységi vizsgálat szintén két részre bontva.

A hiperbolikus programozási feladat esetén a jelentés második része nyolc szakaszból áll. Ez azért van így, mert a "lineáris" jelentésben szereplő négy rész előállítása külön-külön történik a tört-alakú cél-függvény számlálójára és nevezőjére vetítve.

3.3. Az optimális megoldás értelmezése

A megkeresett optimális megoldásról szóló jelentésnek több alkotóeleme van. Ebben az alfelezetben tekintsük részletesen ezeket az elemeket.

3.3.1. *"Activity"*. A bal oldali első két ("No" és "Name") oszlopban találhatjuk a változó numerikus sorszámát és szöveges azonosítóját. Majd

ezeket követi az egykarakteres "Status", amely a változó optimális megoldásban való állapotát jelképezi. A "Status" lehetséges értékei: **A**|**M**|**D**|**Z**.

- **A** mint "**A**ctive" azt jelenti, hogy az adott változó "aktív", azaz szerepel az optimális megoldás bázisában, és az értéke nagyobb nullánál.
- **Z** mint "**Z**ero" azt mutatja, hogy az adott változó nincsen a bázisban, és az értéke nulla.
- **M** mint "**M**ultiple" – ez az érték abban az esetben jelenik meg, ha az adott feladatnak több optimális megoldása van, és az adott változó nem szerepel a feltüntetett optimális megoldás bázisában, de nincs kizárva, hogy egy másik optimális megoldásnál az adott változó a bázisban van.
- **D** mint "**D**egenerate" jelöli a degenerált (azaz nulla értékű) bázis változót.

Az egykarakteres "Status" oszlopot követi a "Value" cimkével ellátott, a primál változó optimális értékét tartalmazó oszlop. A "Value" oszlop után következik a "Shadow costs" oszlop. A lineáris programozási feladat esetén az ebben az oszlopban feltüntetett Δ_j értékek két módon értelmezhetőek:

(1) Ha az adott változó nincsen az optimális bázisban, akkor az adott változóhoz tartozó Δ_j érték azt mutatja, hogy mennyivel rosszabbodik a cél-függvény optimális értéke, ha a változó értéke egy egységgel nő (a mostani nullához képest).
(2) Abban az esetben, ha a változó nincsen a bázisban, akkor Δ_j érték azt mutatja, hogy hány egységgel kellene "javítani" a változóhoz tartozó p_j együttható értékét ahhoz, hogy az adott változó bázisba kerüljön.

A hiperbolikus programozási feladatoknál ilyenkor három "Shadow cost" oszlop jelenik meg: az egyik a cél-függvény számlálójához tartozik, a második a cél-függvény nevezője számára kiszámolt Δ_j'' értékeket tartalmaz, a harmadik pedig a $\Delta_j(x^*)$ értékeket tartalmazza. Mindhárom oszlopban a feltüntett értékeket értelmezhetjük a 1. pontban leírtaknak megfelelően.

3.3.2. *"Constraints".* Ebben a részben az első két oszlop a feladatban szereplő feltételekhez tartozó numerikus sorszámot és szöveges azonosítót tartalmazza. Majd következik a "Slack" oszlop. Az ebben az oszlopban feltüntett értékek nem mások, mint a különbség a feltétel bal és jobb oldala között. Ebből az értékből derül ki, hogy a feltétel jobb oldalán szereplő erőforrás készletéből az optimális megoldásnál marad-e felesleg. A "Slack" oszlopot követő "Shadow price" oszlop duális változókat tartalmaz, amelyek arra a kérdésre adnak a választ, hogy mennyivel javul a cél-függvény optimális értéke abban az esetben, ha a megfelelő erőforrás készletét egy egységgel növeljuk.

3.3.3. *"Stability for Objective".* Az ebben a szakaszban feltüntett jobb oldali három oszlop tartalmazza a cél-függvényben szereplő p_j együtthatók számára összeállított stabilitási tartományt: a tartomány alsó határa ("Lower Limit"), majd maga az aktuális p_j érték, és ezt követi a tartomány felső határa ("Upper Limit"). Ez a tartomány biztosítja az aktuális optimális bázis optimalitását abban az esetben, ha a p_j érték változása a feltüntetett tartományon belül történik.

3.3.4. *"Stability for constraints".* A jelen szakaszban feltüntett jobb oldali három oszlop tartalmazza a feltételekben szereplő b_i értékek számára összeállított stabilitási tartományt: a tartomány alsó határa ("Lower Limit"), majd maga az aktuális b_i érték, és ezt követi a tartomány felső határa ("Upper Limit"). Ez a tartomány biztosítja az aktuális optimális bázis optimalitását abban az esetben, ha a b_i érték változása a feltüntetett tartományon belül történik.

3.4. LP példa

Tekintsünk egy numerikus LP példát. Tegyük fel, hogy egy sertéstenyésztő farmer olyan összetételű takarmányt szeretne összeállítani, amely egyrészt legyen minél olcsóbb, másrészt elegítse ki az állatok összes táplálkozási igényeit. A táplálkozási igényeket (egy állatra vetítve) a 15. táblázatban foglaltuk össze. A minimális igényeknek a "nagyobb mint" relációjú feltételek, a maximális igényeknek "kisebb mint" relációjú feltételek felelnek meg. A farmer rendelkezésére álló alaptakarmányok, illetve az alaptakarmányok táplalkozási tulajdonságai a 16. táblában vannak felsorolva.

Crude protein (**CP**) (nyers protein) %	min	l6.00
Digestible energy (DE) MJ/kg	min	l2.50
	max	l3.50
Lysine (**Ly**) %	min	0.64
Phosphorus (**P**) %	min	0.54
Calcium (**Ca**) %	min	0.72
	max	2.00

15. ábra. WinGULF – Táplálkozási igények.

Manuálisan (azaz a megfelelő szoftvereszközök használata nélkül) ezt a problémát nagyon nehéz lenne megoldani. WinGULF-ban pedig ez a feladat

	Wheat	Lupins	Meat-meal	Dicalcium phosphate	Lime-stone	Lysine
Cost $/Ton	120.00	100.00	325.00	600.00	80.00	3800.0
CP %	11.00	28.00	50.00			
DE MJ/kg	14.60	14.20	11.50			
Lysine %	0.32	0.85	1.53		78.00	
Phosphorus %	0.26	0.29	4.80	22.30		
Calcium %	0.04	0.21	10.00	29.00	33.58	

16. ábra. WinGULF – Rendelkezésre álló alaptakarmányok.

megoldásának keresése nagyon egyszerű folyamattá válik. Először meg kell fogalmazni a feladatnak megfelelő LP modellt, majd ezt az LP modellt be kell gépelni a WinGULF-ban. Mivel a modellalkotási technikák targyalása nem tartozik a jelen jegyzet témaköréhez, ezért feltételezzük, hogy a modell már megvan (az optimálizálási modellek felépítéséről, alkotásáról, megfelelő technikákról lásd.[**55**], [**93**], [**180**], [**189**], [**190**]). A modell begépelése (az adott feladat megtalálható a WinGULF telepítési csomaghoz csatolt *Example1.GLF* nevű állományban) után kapjuk a 17. ábran feltüntett LP feladatot.

LP example		Limit	Wheat	Lupins	MeatMl	Ca2P	LimeSt	Lysine
Cost	N		0.12	0.10	0.325	0.6	0.08	3.8
Obj.Denom	N	1						
CP min(kg)	G	16	0.11	0.28	0.50			
DE min(MJ)	G	1250	14.60	14.20	11.50			
DE max(MJ)	L	1350	14.60	14.20	11.50			
Ly min (g)	G	640	3.20	8.50	15.30			780.0
P min (g)	G	540	2.60	2.90	48.00	223.0		
Ca min (g)	G	720	0.40	2.10	100.00	290.0	335.8	
Ca max (g)	L	2000	0.40	2.10	100.00	290.0	335.8	
Mass (kg)	E	100	1.00	1.00	1.00	1.0	1.0	1.00

17. ábra. WinGULF – LP feladat a WinGULF-ban.

A feladat megoldása után a 18. ábran feltüntett jelentést kapjuk.

Tekintsük át röviden a megjelent jelentés különböző részeit.

Először nézzük az "Optimal value" cimkével ellátott értéket a jelentés "Activities" elötti sorában. Ez a megkeresett legolcsóbb összetételű takarmány ára (10.5 cent per kilogramm). Most akkor következzen az "Activities" rész. Ez a rész a különböző alaptakarmányok mennyiségét tartalmazza az

Optimal Solution

Problem name	: LP example
Problem direction	: MIN
Objective function value	: 10.509784
Number of iterations	: 4

Activities

No	Name		Level	Shad.Cost	LowerObj	Obj	UpperObj
1	Wheat	Z	0.0000	0.0206	0.10	0.120	INF
2	Lupins	A	94.4722	0.0000	0.09	0.100	0.12
3	MeatMl	Z	0.0000	0.1241	0.20	0.325	INF
4	Ca2P	A	1.1930	0.0000	0.08	0.600	1.20
5	LimeSt	A	4.3349	0.0000	-5.24	0.080	0.09
6	Lysine	Z	0.0000	3.7067	0.09	3.800	INF

Constraints

No	Name		Slack	Shad.Price	LowerLim	Limit	UpperLim
1	CP min(kg)	G	10.4522	0.0000	-INF	16.0	26.45
2	DE min(MJ)	G	91.5048	0.0000	-INF	1250.0	1341.50
3	DE max(MJ)	L	8.4952	0.0000	1341.5048	1350.0	INF
4	Ly min (g)	G	163.0134	0.0000	-INF	640.0	803.01
5	P min (g)	G	0.0000	-0.0023	274.4441	540.0	1658.45
6	Ca min (g)	G	1280.0000	0.0000	-INF	720.0	2000.00
7	Ca max (g)	L	0.0000	0.0000	1800.7186	2000.0	4146.52
8	Mass (kg)	E	0.0000	-0.0933	93.6077	100.0	100.59

18. ábra. WinGULF – Optimális megoldási jelentés.

összeállítandó optimális takarmányban. Ezek szerint az optimális (legolcsobb) összetételű takarmány 1 kilója a következő alaptakarmányokból áll: 94.5% lupins, 1.2% Dicalcium Phosphate és 4.3% Limestone.

A "Value" után következő "Shadow Cost" oszlopban feltüntetett Δ_j értékek azt mutatják, hogy mennyivel kell megváltoztatni a megfelelő p_j együtthatót ahhoz, hogy az adott változó a bázisba kerüljön. Például a "Lupins" váltózó már a bázisban van, ezért a hozzá tartozó $p_2 = 0.10$ értéket nem kell változtatni, éppen ezért $\Delta_2 = 0.0$. Viszont ha a "Wheat" (búza) alaptakarmány ára legalább $\Delta_1 = 0.0206$ egységgel olcsóbb lett volna, akkor ez az alaptakarmány már szerepelt volna az optimális megoldásban. Más szavakkal, addig erre az alaptakarmányra nincs szükségünk, amig az ára nagyobb, mint

$$0.12 \text{ cent} \; - \; 0.0206 \text{ cent} \; = \; 0.0994 \text{ cent}$$

per kilogramm.

3.5. HP Példa

Ebben a szekcióban egy hiperbolikus programozási numerikus példával fogunk foglalkozni.

Tegyük fel, hogy egy hűtőszekrényeket gyártó gyárban jelenleg a következő modelleket lehet gyártani: Lehel 220, Lehel 120, Star 200, Star 160 és Star 250. Egy nagykereskedőtől kapott megrendelés szerint a nagykereskedőnek szüksége lenne 150 darab Star 200 készülékre, 70 darab Star 160 készülékre és 290 darab Star 250 készülékre. Ezenkívül még 240 darab tetszőleges készülékre lenne szüksége. Feladatunk abban áll, hogy elő kell állítanunk olyan gyártási tervet, amely kielégíti a nagykereskedő igényeit, ezenkívül biztosítja a gyár számára a legmagasabb "profit/költség" hatékonyságot.

Annak ellenére, hogy a feladatban szereplő hűtőszekrények "egészértékűek", ebben a példában csak folytonos változókkal dolgozunk. Később, a(z) 4. alfejezetben visszatérünk ehhez a feladathoz és akkor a hűtőszekrények "egészértékűek" lesznek.

Az egyszerűség kedvéért ebben a feladatban csak két korlátozott mennyiségben használható erőforrást fogunk figyelembe venni – ezek az ún. "Freon 12" és "TL 16". A feladathoz tartoznak a következő adatok:

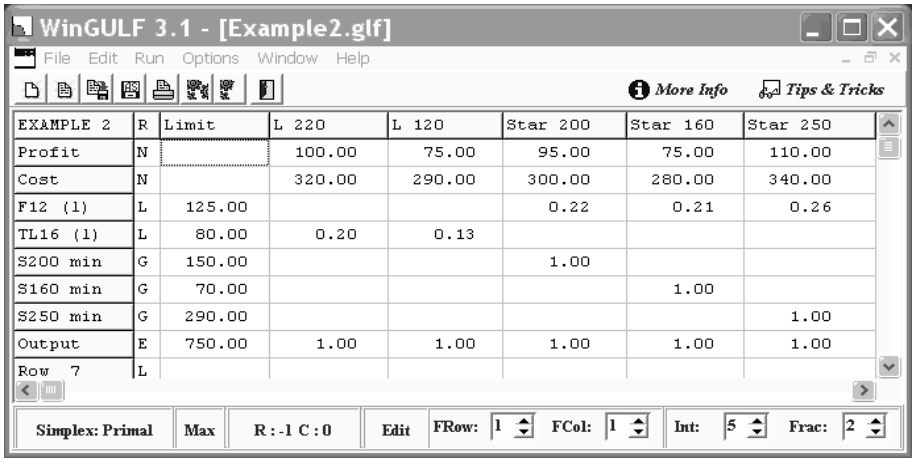
	Lehel 200	Lehel 120	Star 200	Star 160	Star 250
TL 16(l/unit)	0.20	0.13			
F 12(l/unit)			0.22	0.21	0.26
Price $/unit	420.00	365.00	395.00	355.00	450.00
Cost $/unit	320.00	290.00	300.00	280.00	340.00

A feladathoz összeállított mátrix megtekinthető a(z) 19. ábrán. Az ada-

LFP example		Limit	L 220	L 120	S 200	S 160	S 250
Profit $	N		100.00	75.00	95.00	75.00	110.00
Cost $	N		320.00	290.00	300.00	280.00	340.00
F12 (l)	L	125.00			0.22	0.21	0.26
TL16 (l)	L	80.00	0.20	0.13			
S200 min	G	150.00			1.00		
S160 min	G	70.00				1.00	
S250 min	G	290.00					1.00
Output	E	750.00	1.00	1.00	1.00	1.00	1.00

19. ábra. WinGULF – Mátrix a hiperbolikus programozási példához.

tok begépelése (az adott feladat megtalálható a WinGULF telepítési csomaghoz csatolt *Example2.GLF* nevű állományban) után kapjuk a 20. ábran feltüntett HP feladatot. A feladat megoldása után a WinGULF által leg-



20. ábra. WinGULF – HP példafeladat.

enerált jelentés megtekinthető a(z) 21. és 22. ábrán.

Optimal Solution

Problem name : LFP example
Problem direction : MAX
Objective function value : 75473.076923/240146.153846 = 0.314280
Number of iterations : 4

Activities - Numerator

No	Name		Level	Shadow Cost	Lower Obj	Obj	Upper Obj
1	L 220	A	232.69	0.00	84.2689	100.00	100.8255
2	L 120	Z	0.00	25.00	-INFINITY	75.00	90.5716
3	S 200	A	150.00	0.00	92.1506	95.00	96.8155
4	S 160	A	70.00	0.00	68.8942	75.00	90.1804
5	S 250	A	297.31	0.00	108.5624	110.00	529.2583

Activities - Denominator

No	Name		Level	Shadow Cost	Lower Obj	Obj	Upper Obj
1	L 220	A	232.69	0.00	317.3801	320.00	372.6060
2	L 120	Z	0.00	30.00	240.4530	290.00	INFINITY
3	S 200	A	150.00	0.00	294.2441	300.00	309.1179
4	S 160	A	70.00	0.00	232.3684	280.00	299.5385
5	S 250	A	297.31	0.00	-163.1100	340.00	344.6003

21. ábra. WinGULF – HP példafeladat jelentése, 1.rész.

Constraints - Numerator

No	Name		Slack	Shadow Price	Lower Lim	Limit	Upper Lim
1	F12 (l)	L	0.00	38.46	123.1000	125.0	185.5000
2	TL16 (l)	L	33.46	0.00	46.5385	80.0	INFINITY
3	S200 min	G	0.00	-13.46	0.0000	150.0	158.6364
4	S160 min	G	0.00	-33.08	0.0000	70.0	79.0476
5	S250 min	G	7.31	0.00	-INFINITY	290.0	297.3077
6	Output	E	0.00	100.00	517.3077	750.0	917.3077

Constraints - Denominator

No	Name		Slack	Shadow Price	Lower Lim	Limit	Upper Lim
1	F12 (l)	L	0.00	76.92	123.1000	125.0	185.5000
2	TL16 (l)	L	33.46	0.00	46.5385	80.0	INFINITY
3	S200 min	G	0.00	-36.92	0.0000	150.0	158.6364
4	S160 min	G	0.00	-56.15	0.0000	70.0	79.0476
5	S250 min	G	7.31	0.00	-INFINITY	290.0	297.3077
6	Output	E	0.00	320.00	517.3077	750.0	917.3077

22. ábra. WinGULF – HP példafeladat jelentése, 2.rész.

Tekintsük át röviden a kapott jelentést. Először a cél-függvény optimális értéke adódik a számláló (profit) és a nevező (költség) optimális értékéből (lásd. 21. ábra):

$$75473.076923 \,/\, 240146.153846 \;=\; 0.314280.$$

Most nézzük az "Activities" cimke alatt található "Level" oszlopot. Ahhoz, hogy a feladatban rögzített feltételekhez képest a "profit/költség" fajlagos gazdasági mutató maximális értékű legyen, ahhoz a szóban forgó gyártónak elő kell állítania 232.69 darab Lehel 220 készüléket, 150 darab Star 200 készüléket, 70 darab Star 160 készüléket és 297.31 darab Star 250-t. Lehel 120 készülék az optimális gyártási tervben nem szerepel. Nyilvánvaló, a kapott optimális megoldás nem hasznosítható a gyakorlatban mivel tartalmaz nem-egészértékű gyártandó mennyiségeket.

Térjünk át a "Shadow cost" oszlopokhoz (lásd. 21. és 22. ábra). Ezekben az oszlopokban majdnem minden Δ_j nulla értékű. Csak Lehel 120 változóhoz tartozó Δ_2' és Δ_2'' különböznek a nullától. A "Shadow Cost" oszlopokban szereplő "25.00" és "30.00" értékek azt mutatják, hogy a Lehel 120 készüléket akkor érdemes gyártani, ha az egy egysége után járó profit a mostani 75.00 egységhez képest legalább 25.00 egységgel lesz nagyobb, vagy ha az egy egységével járó költség a mostani 290.00 egységhez képest legalább 30.00 egységgel lesz kisebb. Más szavakkal, Lehel 120 készüléket akkor lesz

érdemes gyártani, ha az egy készülékre eső profit legalább

$$75.00\$ + 25.00\$ = 100.00\$$$

lesz, vagy ha az adott készülék egy egységével járó költség nem nagyobb lesz, mint

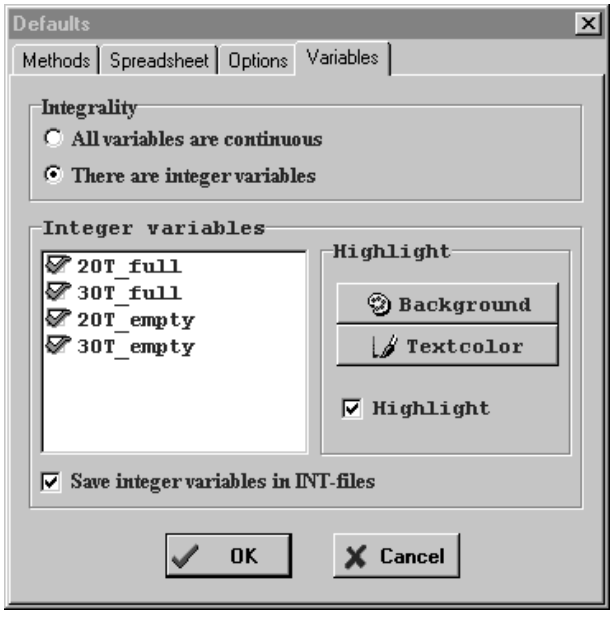
$$290.00\$ - 30.00\$ = 260.00\$$$

4. Egészértékű feladatok

Ebben a szekcióban egészértékű LP és HP feldatokkal fogunk foglalkozni, azaz olyan feladatokkal, amelyekben van legalább egy egészértékű változó.

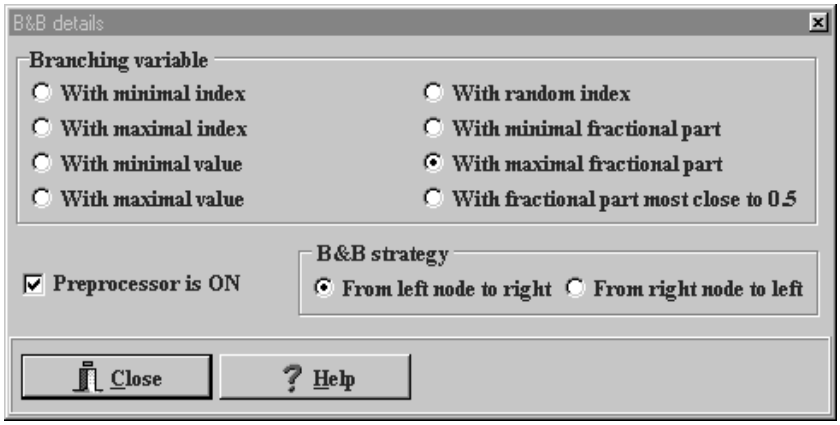
4.1. Bevétel és fő Opciók

Ha a megoldandó feladatban vannak egészértékű változók, akkor ezt a feladatot ugyanúgy be kell gépelni, mint a folytonos esetben. A különbség csak abban áll, hogy utána következik az "Options" párbeszéd ablak a "Variables" lapján történő kijelölése azoknak a változóknak, amelyek csak egész számokat vehetnek értékül. (lásd. 23. ábra). Jelenleg a WinGULF-ban a



23. ábra. WinGULF – Defaults, "Variables" lap.

korlátozás-és-szétválasztási módszer van implementálva. A módszerhez tartozó testreszabási opciókat megtekinthetjük a(z) 24. ábrán. A megoldási



24. ábra. WinGULF – "Branch-and-Bound" módszer, Opciók.

folyamat során összeállítandó bináris fában történő kereséskor a WinGULF programcsomag jelenleg az ún. "először lefelé" keresési szabályt használja, és ezért az opciókban csak két bejárási szabályból választhatunk: "*from left to right*" vagy "*from right to left*". Viszonylag nagyobb és bonyolultabb feladatok esetén gyakran érdemes bekapcsolni a *"Preprocessor"* opciót. Ilyenkor a WinGULF redundáns feltételek keresését végzi, és ha megtalálja azokat, akkor kizárja a feladatból a felesleges feltételeket. Például a következő

$$x_1 \leq 12, \quad \text{és} \quad x_1 \leq 8$$

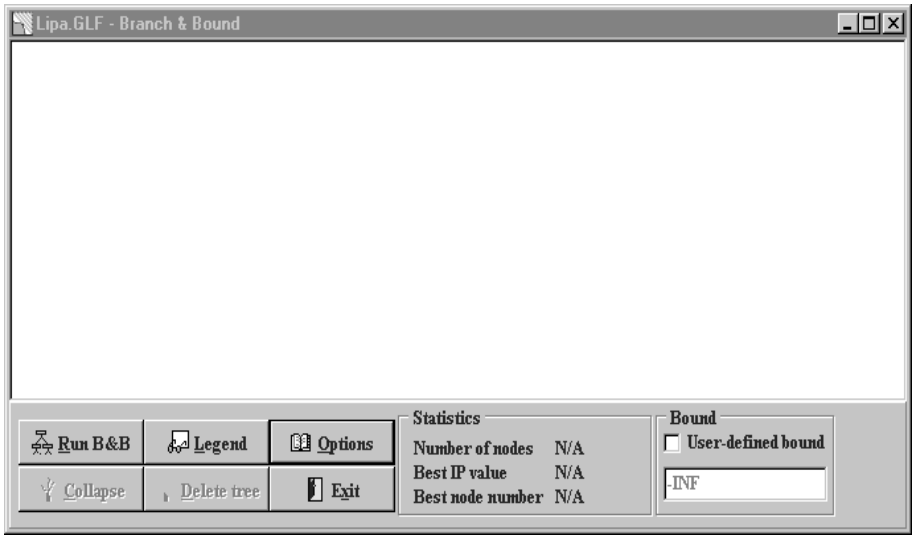
két feltétel helyett a bekapcsolt *"Preprocessor"*-nál a csomópontban megoldandó feladatban csak az utolsó feltétel marad.

4.2. Kimenet

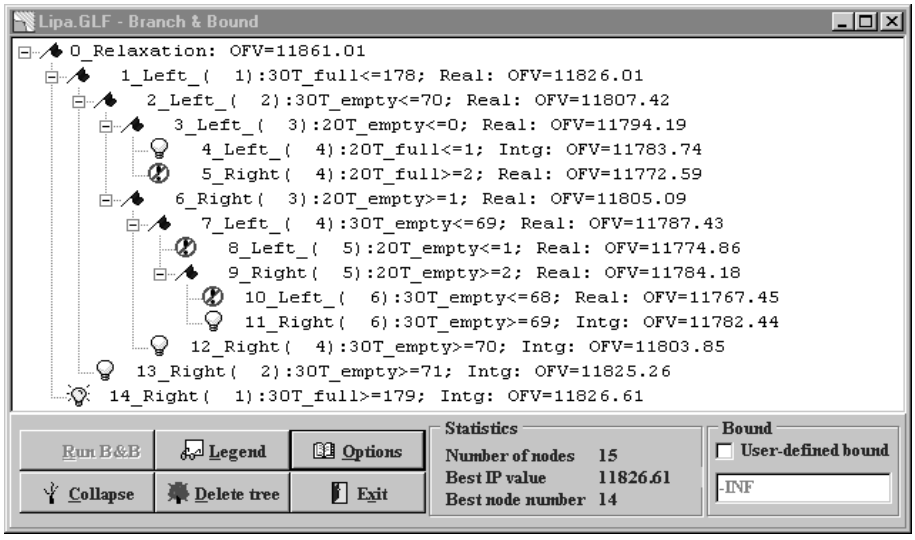
Ha befejeztük a feladat szerkesztését, és már kijelöltük az egészértékű változókat, akkor a *Run* gomb segitségével indíthatjuk a megoldási eljárást. Ekkor a WinGULF megjeleníti a(z) 25. ábrán ábrázolt ablakot. Ha ebben az ablakban a *Run B&B* gombon kattintunk, akkor elindul a "Branch-and-Bound" eljárás. A megoldási eljárás sikeres befejeződése után megjelenik a a(z) 26. ábrán feltüntetett bináris fa. A WinGULF által legenerált jelentés megtekinthető a(z) 27. ábrán.

A szöveges jelentés első részében találhatjuk az igénybe vett beállításokat, majd egy négyoszlopos rész következik, amely a "Branch-and-Bound" módszer végrehajtási folyamatának leírását tartalmazza. Ezek az adatok a következők:

(1) "Node/Level" – a csomópont sorszáma és pozíciója a bináris fában a
<sorszám>/<szint><jobb/bal> formátumban,

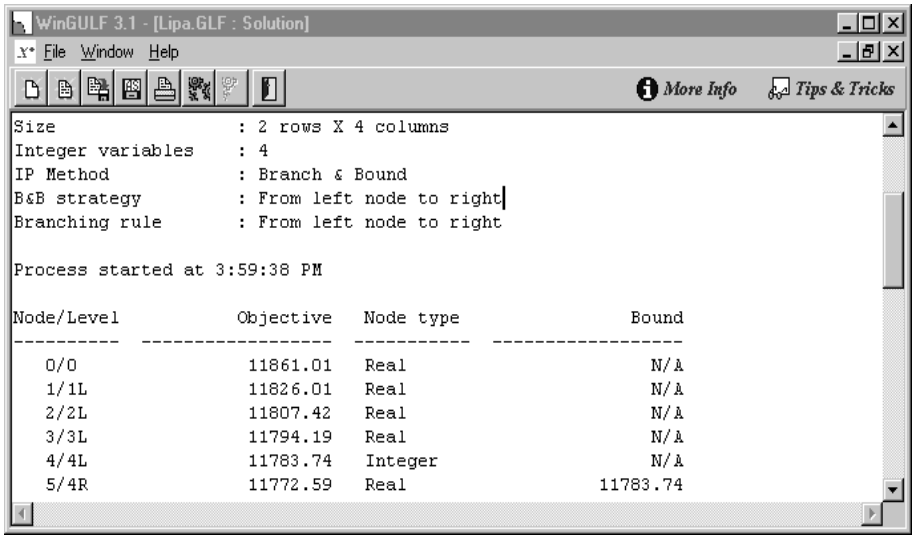


25. ábra. WinGULF – Branch-and-Bound Method, starting.



26. ábra. WinGULF – Branch-and-Bound módszer, grafikus jelentés.

(2) "Objective" – a cél-függvény értéke az adott csomópontban,
(3) "Node type" – a csomópont típusa: "Real"-folytonos megoldás, "Integer"-egészértékű megoldás, "Infeasible"-feladat nem megoldható,
(4) "Bound" – az addig elért legjobb cél-függvény érték.

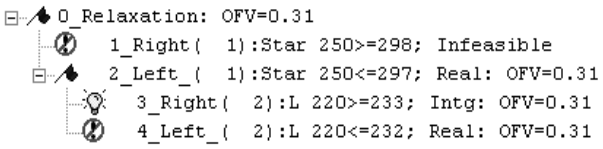


27. ábra. WinGULF – Branch-and-Bound módszer, szöveges jelentés.

4.3. Egészértékű példa

Ebben a szekcióban térjünk vissza a "hűtőszekrényes" hiperbolikus programozási feladathoz. Amikor a fenti részben tanulmányoztuk ezt a feladatot, szándékosan figyelmen kívül hagytuk azt a tényt, hogy a hűtőszekrények gyártandó mennyisége nem tartalmazhat törtrészt. Most nyissuk meg az "*Options*" párbeszéd ablak "*Variables*" lapját, és állítsuk át a feladatban szereplő változókat egészértékű állapotba.

A feladat egészértékű megoldását kereső folyamathoz tartozó bináris fát megtekinthetjük a(z) 28. ábrán.



28. ábra. WinGULF – Bináris fa.

A kapott egészértékű megoldásról szóló jelentést megtekinthetjük a(z) 29. ábrán. A WinGULF által összeállított jeletésből látszik, hogy az összes változó optimális értéke egész.

```
Node/Level           Objective   Node type          Bound
----------  ------------------  -----------  -------------
   0/0                    0.31   Real                 N/A
   1/1R                    N/A   Infeasible           N/A
   2/1L                   0.31   Real                 N/A
   3/2R                   0.31   Integer              N/A
   4/2L                   0.31   Real                0.31

Optimal solution found at 11:02:33 AM in 00h.00m.00s.531ms.
Optimal value        : 0.31427501
Number of nodes      : 5
Best node number     : 3

Activities
----------
 No Name                            Value
--- ---------- - ------------------------
  1 L 220      A                   233.00
  2 L 120      Z                     0.00
  3 Star 200   A                   150.00
  4 Star 160   A                    70.00
  5 Star 250   A                   297.00

Constraints
-----------
 No Name                            Slack
--- ---------- - ------------------------
  1 F12 (l)    L                     0.08
  2 TL16 (l)   L                    33.40
  3 S200 min   G                     0.00
  4 S160 min   G                     0.00
  5 S250 min   G                     7.00
  6 Output     E                     0.00
```

29. ábra. WinGULF – Optimális jelentés.

Irodalomjegyzék

"Knowledge is of two kinds.
We know a subject ourselves,
or we know where we can find
information upon it"
– Samuel Johnson, 1775

[1] Abrham,J., Luthra,S., *"Comparison of Duality Models in Fractional Linear Programming"*, Zeitschrift für Operations Research, Vol.21, 1977, pp.125-130.

[2] Aggarwal,S.P., *"Analysis of the solution to a linear fractional functionals programming"*, Metrika, Vol.16, 1970, pp.9-26.

[3] Almogy,Y., Levin,O., *"A Class of Fractional Programming Problems"*, Operations Research, Vol.19, 1971, pp.57-67.

[4] Arora,S.R., Puri,M.C., *"Enumeration Technique for the Set Covering Problem with Linear Fractional Functional as its Objective Function"*, Zeitschrift für Angewandte Mathematik und Mechanik, Vol.56, 1977, pp.181-186.

[5] Arora,S.R., Puri,M.C., Swarup,K., *"The Set Covering Problem with Linear Fractional Functional"*, Indian Journal of Pure and Applied Mathematics, Vol.8, No.5, 1977, pp.578-588.

[6] Arvalo,M.T., Mrmol,A.M., Zapata,A. *"The Tolerance Approach in Multiobjective Linear Fractional Programming"*, Sociedad de Estadística e Investgación Operativa, Vol.5, No.2, 1997, pp.241-253.

[7] Ashton,D.J., Atkins,D.R., *"Multi-criteria Programming for Financial Planning"*, Journal of the Operational Research Society, Vol.30, 1979, pp.259-270.

[8] Bajalinov,E.B., *"Duality in Linear-Fractional Programming and its Applications"*, Ph.D. Thesis, Institute of Mathematics, Kirghiz Academy of Sciences, Frunze, Kirghiz Republic, 1984. (in Russian)

[9] Bajalinov,E.B., *"On the Economic Sense of Dual Variables in Linear-Fractional Programming"*, Ekonomika i matematicheskie metody, 3, Vol.24, 1988, pp.558-561. (in Russian)

[10] Bajalinov,E.B., *"On the Coincidence of the Optimal Solutions in Linear and Linear Fractional Programming Problems"*, Izvestia Akademii Nauk Kirgizskoi SSR, No.3, 1988. (in Russian)

[11] Bajalinov,E.B., *"On the System of Three Problems of Mathematical Programming"*, Kibernetika, No.6, 1989. (in Russian)

[12] Bajalinov,E.B., *"On Concordance of the Economic Interests"*, Izvestia Akademii Nauk Kirgizskoi SSR, No.3, 1990. (in Russian)

[13] Bajalinov,E.B., *"On an Approach to the Modelling of Problems Connected with Conflicting Economic Interests"*, European Journal of Operational Research, Vol.116, 1999, pp. 477-486.

[14] Bajalinov,E.B., *"Linear-Fractionl Programming: Theory, Methods, Applications and Software"*, Kluwer Academic Publishers, 2003.

[15] Bajalinov,E.B., Imreh,B., *"Operációkutatás"*, Polygon jegyzettár, Szeged, 2001.

[16] Bajalinov,E.B., Pannel,D.J., *"GULF : a General, User-friendly Linear and linear – Fractional programming package"*, Technical Report No. 93/86, Department of Mathematics, University of L.Kossuth, Debrecen, Hungary, 1993.

[17] Balas,E., *"An Additive Algorithm for Solving Linear Programs with Zero-One Variables"*, Operations Research, Vol.13, 1965, pp.517-546.

[18] Balas,E.,Ceria,S., Cornujols,G., *"A Lift-and-Project Cutting Plane Algorithm for Mixed 0/1 Programs"*, Mathematical Programming, Vol.58, 1993, pp.295-324.

[19] Balas,E.,Ceria,S., Cornujols,G., Natraj,N., *"Comory Cuts Revised"*, Operations Research Letters, Vol.19, 1996, pp.1-9.

[20] Barros,A.I., *"Discrete and Fractional Programming Techniques for Locations Models"*, Seria "Combinatorial Optimization", Vol. 3, Kluwer Academic Publishers, 1998.

[21] Barros,A.I., Frenk,J.B.G., Schaible,S., Zhang,S., *"A New Algorithm for Generalized Fractional Programs"*, Mathematical Programming, Vol.72, 1996, pp.147-173.

[22] Bartels,R.H., Golub,G.H., *"The Simplex Method of Linear Programming Using LU-Decomposition"*, Communication of the ACM, Vol.12, 1969, pp. 266-268 and 275-278.

[23] Beale,E.M.L., Small,R.E., *"Mixed Integer Programming by a Branch and Bound Technique"*, Proc. IFIP. Congr. 2, 1965, pp.450-451.

[24] Beasley,J.E., *"Advances in Linear and Integer Programming"*, Oxford Lecture Series in Mathematics and Its Applications, Vol.4, Oxfors University Press, 1996.

[25] Bector,C.R., *"Duality in Fractional and Indefinite Programming"*, Zeitschrift für Angewandte Mathematik und Mechanik, Vol.48, No.6, 1968, pp.418-420.

[26] Bector,C.R., *"Duality in Linear Fractional Programming"*, Utilitas Mathematica, Winnipeg, Vol.4, 1973, pp.155-168.

[27] Bector,C.R., *"Duality in Nonlinear Fractional Programming"*, Zeitschrift für Operations Research, Vol.17, 1973, pp.183-193.

[28] Bector,C.R., Chandra,S., Singh,C., *"Duality in Multiobjective Fractional Programming"*, in "International Workshop on Generalized Concavity, Fractional Programming and Economic Applications", University of Pisa, Italy, 1989.

[29] Belykh,V.M., Gavurin,M.K., *"An Algorithm for Minimizing a Fractional-Linear Function"*, Bulletin of the Leningrad State University, Vol.19, No.4, Oct.1980, pp.10-15. (in Russian)

[30] Bitran,G.R., Magnanti,T.L., *"Fractional Programming: Duality, Algorithms, Sensitivity Analysis and Applications"*, Technical Report No.92, Operations Research Center, Massachusets Institute of Technology, June 1974.

[31] Bitran,G.R., Magnanti,T.L., *"Duality and Sensitivity Analysis for Fractional Programs"*, Operations Research, No.4, Vol.24, 1976, pp.675-699.

[32] Bixby, R.E., *"Implementing the Simplex Method: The Initial Basis"*, ORSA J. on Computing, Vol.4, No.3, pp. 267-284, Summer, 1992.

[33] Bland,R., *"New Finite Pivoting Rules for the Simplex Method"*, Mathematics of Operations Research, Vol.2, 1977, pp. 103-107.

[34] Borde,J., Crouzeix,J.-P., *"Convergence of a Dinkelbach-type Algorithm in Generalized Fractional Programming"*, Zeitschrift für Operations Research, Vol.32, 1987, pp.A31-A54.

[35] Cambini,A., Martein,L., Schaible,S., *"On Maximizing a Sum of Ratios"*, Journal of Information and Optimization Sciences, Vol.10, 1989, pp.65-79.

[36] Ceria,S., Cornujols,G., Dawande,M. *"Combining and Strengthening Gomory Cuts"*, in Balas,E., Clausen,J., (eds.), Lecture Notes in Computer Science, Vol.920, Spinger-Verlag, 1995.

[37] Chandra,S, Chandramohan,M., *"An Improved Branch and Bound Method for Mixed Integer Linear Fractional Program"*, Zeitschrift für Angewandte Mathematik und Mechanik, Vol.59, No.10, 1979, pp.575-577.

[38] Chandra,S, Chandramohan,M., *"A Note on Integer Linear Fractional Program"*, Naval Research Logistics Quarterly, Vol.27, 1980, pp.171-174.

[39] Chandrasekaran,R., *"Minimal Ratio Spanning Trees"*, Networks, Vol.7, 1977, pp.335-342.

[40] Charnes,A., Cooper,W.W., *"Programming with Linear Fractional Functionals"*, Naval Res. Logistics Quart., Vol.9, No.3 and 4, 1962, pp.181-186.

[41] Chadha,S.S., *"Dual Fractional Program"*, ZAMM, Vol.51, 1971, pp.560-561.

[42] Chernov,Y.P., Lange,E.G., *"Problems of Nonlinear Programming with Fractional Economic Criteria. Methods and Applications"*, Kirghiz Academy of Science, Ilim, Frunze, 1978. (in Russian)

[43] Choo,E.U., Atkins,D.R., *"Bicriteria Linear Fractional Programming"*, Journal of Optimization Theory and Applications, Vol.36, 1982, pp.203-220.

[44] Chvátal,V., *"Linear Programming"*, Freeman, New York, 1983.

[45] Craven,B.D., Mond,B., *"The Dual of a Fractional Linear Program"*, Journal of Mathematical Analysis and Applications, Vol.42, 1973, pp. 507-512.

[46] Crouzeix,J.-P., Ferland,J.A., *"Algorithms for Generalized Fractional Programming"*, Mathematical Programming, Vol.52, 1991, pp.191-207.

[47] Crouzeix,J.-P., Ferland,J.A., Schaible,S., *"Duality in Generalized Linear Fractional Programming"*, Mathematical Programming, Vol.27, 1983, pp.342-354.

[48] Crouzeix,J.-P., Ferland,J.A., Schaible,S., *"An Algorithm for Generalized Fractional Programs"*, Journal of Optimization Theory and Applications, Vol.47, 1985, pp.35-49.

[49] Curtis,A.R., Reid,J.K., *"On the Automatic Scaling of Matrices for Gaussian Elimination"*, J. Inst. Maths. Applics., Vol.10, pp.118-124, 1972.

[50] Craven,B.D., *"Fractional Programming"*, Sigma Series in Applied Mathematics, Vol.4, Heldermann Verlag, Berlin, 1988.

[51] Dai,Y., Shi,J., *"A Conical Partition Algorithm for Maximizing the Sum of Several dc Ratios"*, Proceedings of the 5th International Conference on Optimization: Tech. Applications (ISOTA 2001), Hong Kong, 2001, pp.600-608.

[52] Dakin,R.,J., *"A Tree-Search Algorithm for Mixed Integer Programming Problems"*, Computer Journal, Vol.8, 1965, pp.250-255.

[53] Dantzig,G.B., *"Maximization of a Linear Function of Variables Subject to Linear Inequalities"*, In *Activity Analysis of Production and Allocation*, edited by T.C.Koopmans. New-York, John Wiley and Sons, 1951.

[54] Dantzig,G.B., *"Linear Programs and Extensions."* Princeton, New Jersey, Princeton University Press, 1963.

[55] Dent,J.B., Harrison,S.R. and Woodford,K.B., *"Farm Planning With Linear Programming: Concept and Practice"*, Butterworths, Sydney, 1986.

[56] Dinkelbach,W., *"Die Maximierung Eines Quotienten Zweier Linearer Funktionen Unter Linearen Nebenbedingungen"*, Wahrscheinlichkeitstheorie, Vol.1, 1962, pp.141-145.

[57] Dorn,W.S., *"Linear Fractional Programming"*, IBM Research Report RC-830, Yorktown Heights, New York, November 1962.

[58] Duff,I.S., *"A Survey of Sparse Matrix Research"*, Proc. IEEE 65, pp.500-535, 1977.

[59] Duff,I.S., Erisman,A.M., Reid,J.K., *"Direct Methods for Sparse Matrices"*, Clarendon Press, Oxford, 1986.

[60] Dutta,D., Rao,J.R., Tiwari,R.N., *"Fuzzy Approaches for Multiple Criteria Linera Fractional Optimization: a comment"*, Fuzzy Sets and Systems, Vol.54, 1993, pp. 347-349.

[61] Eijkhout,V., *"LAPACK Working Note 50: Distributed Sparse Data Structures for Linear Algebrs Operations"*, Technical Report CS

92-169, Computer Science Department, University of Tennessee, Knoxville, TN, 1992.

[62] Falk,J.E., Palocsay,S.W., *"Optimizing the Sum of Linear Fractional Functions"*, Recent Advances of Global Optimization, Princeton University Press, Princeton 1992, pp.221-258.

[63] Fletcher,R., Matthews,S.P.J., *"Stable Modification of Explicit LU Factors for Simplex Updates"*, Mathematical Programming, Vol.30, 1984, pp.267-284.

[64] Fletcher,R., Matthews,S.P.J., *"A Stable Algorithm for Updating Triangular Factors Under a Rank One Change"*, Mathematics of Computations, Vol.45, No.172, 1985, pp.471-485.

[65] Fletcher,R., *"Practical Methods of Optimization"*, Wiley-Interscience, 1987.

[66] Forrest,J.J.H., Tomlin,J.A., *"Updating Triangular Factors of the Basis Matrix to Maintane Sparsity in the Product Form Simplex Method"*, Mathematical Programming, Vol.2, 1972, pp.263-278.

[67] Freund,R.W., Jarre.F., *"An Interior-Point Method for Convex Fractional Programming"*, AT&T Numerical Analysis Manuscript, No.93-03, Bell Laboratories, Murray Hill, NJ, 1993.

[68] Freund,R.W., Jarre.F., *"An Interior-Point Method for Multi-Fractional Programs with Convex Constraints"*, AT&T Numerical Analysis Manuscript, No.93-07, Bell Laboratories, Murray Hill, NJ, 1993.

[69] Fukuda,K., Terlaky,T., *"Criss-Cross Method: a Fresh View on Pivot Algorithms"*, Mathematical Programming, B79, 1997, pp.369-395.

[70] Gavurin,M.K., *"Fractional-Linear Programming on an Unbounded Set"*, Bulletin of the Leningrad State University, Vol.19, No.4, Oct.1982, pp.12-16. (in Russian)

[71] Gass,S.I., *"Linear programming"*, McGraw-Hill, New-York, 1958.

[72] Gill,P.E., Golub,G.H., Murray,W., Saunders,M.A., *"Methods for Modifying Matrix Factorizations"*, Mathematics of Computations, Vol.28, 1974, pp. 505-535.

[73] Gill,P.E., Murray,W., Saunders,M.A., Wright,M.H., *"Maintaining LU Factors of a General Sparse Matrix"*, Linear Algebra and its Aplications, 1988, pp.239-270.

[74] Gill,P.E., Murray,W., Wright,M.H., *"Numerical Linear Algebra and Optimization"*, Addison-Wesley, 1991.

[75] Glover,F., *"A Multiphase-Dual Algorithm for the Zero-One Integer Programming Problem"*, Operations Research, Vol.13, 1965, pp.879-919.

[76] Glover,F., Laguna,M., *"Tabu Search"*, Kluwer Academic Publishers, 1997.

[77] Goedhart,M.H., Spronk, J., *"Financial Planning with Fractional Goals"*, European Journal of Operational Research, Vol.82, 1995, pp.111-123. North-Holland.

[78] Gol'stein,E.G., *"Dual Problems of Convex and Fractional-Convex Programming in Functional Spaces"*, Doklady Akademii Nauk SSSR, Vol.172, No.5, 1967, pp.1007-1010. (in Russian)

[79] Gol'stein,E.G., *"Duality Theory in Mathematical Programming and its Applications"*, Nauka, Moscow, 1971. (in Russian)

[80] Gol'stein,E.G., Yudin,D.B., *"Linear programming problems of transportation type"*, Nauka, Moscow, 1969. (in Russian)

[81] Golub,G.H., Van Loan,C.F., *"Matrix Computations"*, Baltimore, The Johns Hopkins University Press, 1996.

[82] Gomory,R., *"Outline of an Algorithm for Integer Solutions to Linear Programs"*, Bulletin of the American Mathematical Society, Vol.64, 1958, pp.275-278.

[83] Gomory,R., *"An Algorithm for Integer Solutions to Linear Programs"*, Recent Advances in Mathematical Programming, in Graves,R.L., and Wolfe,P. (eds.), McGraw-Hill, 1963, pp.269-302.

[84] Gondzio,J., *"Stable Algorithm for Updating Dense LU Factorization After Row or Column Exchange and Row and Column Addition or Deletion"*, Optimization, Vol.23, 1992, pp.7-26.

[85] Gondzio,J., *"Applying Schur Complements for Handling General Updates of a Large, Sparse, Unsymmetric Matrix"*, Technical Report ZTSW-2-G244/93, Systems Research Institute, Polish Academy of Sciences, 1995.

[86] Granot,D., Granot,F., *"On Integer and Mixed Integer Fractional Programming Problems"*, Annals of Discrete Mathematics 1, Studies in Integer Programming, (eds.) Hammer,P.L., North Holland Publishing Company, 1977, pp.221-231.

[87] Gupta,B., *"Finding the Set of all Efficient Solutions for the Linear Fractional Multiobjective Program with Zero-One Variables"*, Operations Research, Vol.18, 1981, pp.204-214.

[88] Gupta,R., Malhotra,R., *"Multi-Criteria Integer Linear Fractional Programming Problem"*, Optimization, Vol.35, 1995, pp.373-389.

[89] Gustavson,F.G., *"Some Basic Technigues for Solving Sprase Systems of Linear Equations"*, In "Sparse Matrices and thier Applications", Ed.: Rose,D.J., Willoughby,R.A., Proceedings of Symposium at IBM Research Center, NY, September 9-10, 1971.

[90] Hansen,P., De Aragão,M.V.P., Ribeiro,C.C., *"Hyperbolic* $0-1$ *Programming and Query Optimization in Information Retrieval"*, Mathematical Programming, Vol.52, 1991, pp.255-263.

[91] Hansen,P.,Pedrosa Filho,E.L., Ribeiro,C.C., *"Locations and Sizing of Offshore Platforms for Oil Exploration"*, European Journal of Operational Research, Vol.58(1), pp.202-214, 1992.

[92] Hansen,P.,Pedrosa Filho,E.L., Ribeiro,C.C., *"Modeling Location and Sizing of Offshore Platforms"*, European Journal of Operational Research, Vol.72(3), pp.602-605, 1994.

[93] Hardaker,J.B., *"Farm Planning by Computer", MAFF/ADAS, Reference Book 419*, Her Majesty's Stationery Office, London, 1980.

[94] Hartmann,K., *"Einige Aspekte der Ganzzahligen Linearen Quotientenoptimierung"*, Wiss Z.Tech.Hochsch. Chem.Leuna-Merseburg., Vol.15, No.4, 1973, pp.413-418.

[95] Hartmann,K., *"Zur Anwendung des Schnittverfahrens von Gomory auf Gemischt Ganzzahlige Lineare Quotientenoptimierungsprobleme"*, III Internat. Tagung "Mathematik und Kybernetik in der Ökonomie", 1973.

[96] Hartwig,H., *"Ein Simplexartigen Lösungsalgorithmus für Pseudolineare Optimierungsprobleme"*, Studia Sci. Math. Hungar., Vol.10, No.1-2, 1975, pp.213-236.

[97] Heath,M.T., *"Scientific Computing. An Introductory Survey"*, McGraw-Hill, 2002.

[98] Heesterman,A.R.G., *"Matrices and Simplex Algorithms"*, D.Reidel Publishing Company, 1983.

[99] Hirche,J., *"Optimizing of Sums and Products of Linear Fractional Functions Under Linear Constraints"*, Perprints Series, 95-03.

[100] Hoffman,A.J., Mannos,M., Sokolowsky,D., Weigmann,N., *"Computational Experience in Solving Linear Programms"*, Journal of the Society for Industrial and Applied Mathematics, Vol.1, No.1, 1953, pp.17-33.

[101] Illés,T., Szirmai,Á., Terlaky,T., *"The Finite Criss-cross Method for Hyperbolic Programming"*, European Journal of Operational Research, Vol.114, 1999, pp. 198-214.

[102] Ishii,H., Ibaraki,T., Mine,H., *"A Primal Cutting Plane Algorithm for Integer Fractional Programming Problems"*, Journal of the Operations Research Society of Japan, Vol.19, No.3, 1976, pp.228-244.

[103] Ishii,H., Ibaraki,T., Mine,H., *"Fractional Knapsack Problems"*, Mathematical Programming, Vol.13, 1976, pp.255-271.

[104] Ishii,H., Nishida,T., Daino.A., *"Fractional Set Covering Problems"*, Technical Report No. 1492, Osaka University, Vol. 29, 1979, pp.319-326.

[105] Isbell,J.R., Marlow,W.H., *"Attrition Games"*, Naval Research Logistics Quarterly, Vol.3, 1956, pp.71-94.

[106] Jagannathan,R., Schaible,S., *"Duality in Generalized Fractional Programming via Farkas' Lemma"*, Journal of Optimization Theory and Applications, Vol.41, 1983, pp.417-424.

[107] Jo,C.L., Kim,D.S., Lee,G.M., *"Duality for Multiobjective Fractional Programming Involving n-Set Functions"*, Optimization, Vol.29, 1994, pp.205-213.

[108] Jonathan,S., Kornbluth,H., Steuer,R.E., *"Multiple Objective Linear Fractional Programming"*, Management Science, Vol.27, No.9, 1981, pp.1024-1031.

[109] Kantorovich,L.V., *"Economic Accounting for the Best Utilization of Resources"*, AN SSSR, Moscow, 1960. (in Russian)

[110] Karmarkar,N, *"A new Polynomial-Time Algorithm For Linear Programming"*, Combinatorica, Vol.4, No.4, 1984, pp.373-395.

[111] Kaška,J., *"Duality in linear fractional programming"*, Ekonomicko-Matematicky Obzor, Vol.5, No.4, 1969, pp.442-453.

[112] Kaul,R.N., Bhatia,D., *"Generalized Linear Fractional Programming"*, Ekonomocko-Matematicky Obzor, Vol.10, No.3, 1974, pp.322-330.

[113] Khachian,L.G., *"A Polynomial Algorithm in Linear Programming"*, Doklady Akademii Nauk SSSR, Vol.244, 1979, pp.1093-1096.

[114] Knuth,D.E., *"The Art of Computer Programming"*, Vol.1:*"Fundamental Algorithms"*, Adisson-Wesley, 1968.

[115] Konno,H., Yajima,Y., *"Minimizing and Maximizing the Product of Linear Fractional Functions"*, Recent Advances of Global Optimization, Princeton University Press, Princeton 1992, pp.259-273.

[116] Kornbluth,J.S.H., *"A Survey of Goal Programming"*, OMEGA, Vol.1, 1973, pp.193-205.

[117] Kornbluth,J.S.H., Salkin,G.R., *"A Note on the Economic Interpretation of the Dual Variables in Linear Fractional Programming"*, ZAMM, 52, 1972.

[118] Kornbluth,J.S.H., Steuer,R.E., *"Multiple Objective Linear Fractional Programming"*, Management Science, Vol.9, No. 27, 1981, pp.1024-1039.

[119] Kornbluth,J.S.H., Vinso,J.D., *"Capital Structure and the Financing of Multinational Corporation: A Fractional Multiobjective Approach"*, Journal of Financial and Quantitative Amalysis, Vol.17, 1982, pp.147-178.

[120] Kotiah,T., Slater,N., *"On Two-Server Poisson Queues with Two Types of Customers"*, Operations Research, Vol.21, 1973, pp.597-603.

[121] Kotiah,T., Steinberg,D.I., *"Occurrence of Cycling and Other Phenomena Arising in a Class of Linear Programming Models"*, Commun. ACM, Vol.20, No.2, 1977, pp.102-112.

[122] Kuhn,H.W., *"The Hungarian Method for the Assignment Problem"*, Naval Research Logistics Quaterly, Vol.2, No.2-3, 1955, pp.83-97.

[123] Kuhn,H.W., Quandt,R.E., *"An Experimantal Study of the Simplex Method"*, Proceedings of the Symposia in Applied Mathemetics, Vol.15, American Mathematical Society, 1963.

[124] Kydland,F., *" Duality in fractional programming"*, Naval Research Logistics Quarterly, Vol.19, No.4, 1972, pp.691-697.

[125] Land,A.H., Doig,A.G., *"An Automatic Method of Solving Discrete Programming Problems"*, Econometrica, Vol.28, No.3, 1960, pp.497-520.

[126] Larcombe,M.H.E., *"A List Processing Approach to the Solution of Large Sparse Sets of Matrix Equations and the Factorizations of the Overall Matrices"*, In "Large Sparse Sets of Linear Eqiations", Ed.: Reid,J.K., Academic Press, London, 1971.

[127] Lasdon,L.S., *"Optimization Theory for Large Systems"*, Macmillan, Collier-MacMillan, London, 1970.

[128] Lawler,E.L., Lenstra,J.K., Rinnoy Kan,A.H.G., Shmoys,D.B., (eds.) *"The Traveling Salesman Problem"*, John Wiley & Sons, Ltd., 1985.

[129] Liebling,T.M., *"On the Number of Iterations of the Simplex Method"*, Methods of Operations Research, Vol.17, No.5, Oberwolfach-Tagung uber Operations Research, 13-19, 1977, pp. 248-264.

[130] Luhandjula,M.K., *"Fuzzy Approaches for Multiple Objective Linear Fractional Optimization"*, Fuzzy Sets and Systems, Vol.13, 1984, pp.11-23.

[131] Lutsma,F.A., *"Multi-Criteria Decision Analysis via Ratio and Difference Judgement"*, in Series of Applied Optimization, Vol.29., Kluwer Academic Publishers, 1999,

[132] Magee,T.M., Glover,F., *"Integer Programming: Mathematical Programming for Industrial Engineers"*, Avriel,M., and Golany,B.(eds.), Marcel Dekker, Inc. New York, 1995, pp.123-270.

[133] Martos,B., *"Hyperbolic Programming"*, Publ. Math. Inst., Hungarian Academy of Sciences, Vol.5, ser. B, 1960, pp.386-406.

[134] Martos,B., *"Hyperbolic Programming"*, Naval Research Logistics Quarterly, Vol.11, 1964, pp.135-155.

[135] Matsui, T., Saruwatari,Y., Shigeno,M., *"An Analysis of Dinkelbach's Algorithm for 0-1 Fractional Programming Problems"*, Technical Reports METR92-14, Department of Mathematical Engineering and Information Physics, University of Tokyo, 1992.

[136] Mukherjee,R.N., *"Generalizad Convex Duality for Multiobjective Fractional Programs"*, Journal of Mathematical Analysis and Applications, Vol.162, 1991, pp.309-316.

[137] Murty,K.G., *"Linear Programming"*, John Wiley and Sons, 1983.

[138] Myung,Y., Tcha,D., *"Return of Investment Analysis for Facility Location"*, Technical Reprt OR 251-91, Massachusetts Institute of Technology, 1991.

[139] Nemhauser,G.L., Wolsey,L.A., *"Integer and Combinatorial Optimization"*, John Wiley & Sons, New York, 1988.

[140] Nemirovskii,A.S., Nesterov,Y., *"An Interior-point Method for Generalized Linear-Fractional Programming"*, Mathematical Programming, Vol.69, 1995, pp.177-204.

[141] Nemirovskii,A.S., *"On Polynomiality of the Method of Analytical Centers for Fractional Programming"*, Mathematical Programming, Vol.73, 1996, pp.175-198.

[142] Nemirovskii,A.S., *"The Long-Step Method of Analytical Centers for Fractional Problems"*, Mathematical Programming, Vol.77, 1997, pp.191-224.

[143] Nesterov,Y., Nemirovskii,A.S., *"Interior Point Polynomial Algorithms in Convex Programming: Theory and Applications"*, SIAM, Philadelphia, 1994.

[144] Neumann von,J., *"A Model of General Economic Equilibrium"*, Review of Economic Studies, Vol.13, 1945, pp.1-9.

[145] Nykowski,I., Zolkiewski,Z., *"A Compromise Procedure for the Multiple Objective Linear Fractional Programming Problem"*, European Journal of Operational Research, Vol.19, 1985, pp.91-97.

[146] Orden,A., *"Computaional Investigation and Analysis of Probabilistic Parameters of Convergence of a Simplex Method"*, In Progress in Operations Research, Vol.2, Ed. Prekopa A., North-Holland, Amsterdam, The Netherlands, 1976, pp.705-715.

[147] Parker,G., Radin,R., *"Discrete Optimization"*, Academic Press, New York, 1988.

[148] Pissanetzky,S., *"Sparse Matrix Technology"*, Academic Press, 1984.

[149] Press,W.H., Teukolsky,S.A., Vetterling,W.T., Flannery,B.P., *"Numerical Recipes in C. The Art of Scientific Computing."*, Second Edition, The University of Cambridge, 1992.

[150] Reid,J.K., *"Fortran Subroutines for Handling Sparse Linear Programming Bases"*, HMSO, London, UK, Report AERE-R.8269, 1976.

[151] Reid,J.K., *"A Sparsity Exploiting Variant of the Bartels-Golub decomposition for Linear Programming Bases"*, Mathematical Programming, Vol.24, 1982, pp.55-69.

[152] Rheinboldt,W.C., Mesztenyi,C.K., *"Programs for the Solution of Large Sparse Matrix Problems Based on the Arc-graph Structures"*, Computer Science Center, University of Maryland, College Park, Technical Report TR-262, 1973.

[153] Ritter,K., *"A Parametric Method for Solving Certain Nonconcave Maximization Problems"*, Journal of Computer System Science, Vol.1, 1967, pp.44-54.

[154] Robillard,P., *"(0/1) Hyperbolic Programming Problems"*, Naval Research Logistics Quarterly, Vol.18, 1971, pp.47-57.

[155] Roos,C., Terlaky,T., Vial,J.-Ph., *"Theory and Algorithms for Linear Optimization"*, John Wiley & Sons, 1997.

[156] Rosing,K.E., *"Considering offshore production platforms"*, European Journal of Operational Research, Vol.72(1), pp.204-206, 1994.

[157] Rothenberg,R.I., *"Linear programming"*, North-Holland, 1979.

[158] Saad,O.M., *"An Algorithm for Solving the Linear Fractional Programs"*, Journal of Information & Optimization Science, Vol.14, No.1, 1993, pp.87-93.

[159] Saad,Y., *"SPARSKIT: A Basic Tool Kit for Sparse Matrix Computation"*, Technical Report CSRD TR 1029, CSRD, University of Illinois, Urbana, IL, 1990.

[160] Saunders,M.A. *"A Fast, Stable Implementation of the Simplex Method Using Bartels-Golub Updating"*, In Sparse Matrix Computations, Ed. Bunch,J.R., Rose,D.J., Academis Press, 1976, pp.213-226

[161] Savelsbergh,M.W.P., *"Preprocessing and Probing Techniques for Mixed Integer Programming Problems"*, ORSA Journal on Computing, Vol.6, No.4, 1994, pp.445-454.

[162] Saxena,P.C., Patkar,V.N., Parkash,O., *"A Note on an Algorithm for Integer Solution to Linear and Piecewise Linear Programs"*, Pure and Applied MAthematic Science, Vol.9, No.1-2, 1979, pp.31-36.

[163] Schaible,S., *"Fractional Programming: Transformations, Duality and Algorithmic Aspects"*, Technical Report Vol.73-9, Department of Operations Research, Stanford University, November 1973.

[164] Schaible,S., *"Duality in Fractional Programming: A Unified Approach"*, Operations Research, Vol.24, No.3, 1976, pp.452-461.

[165] Schaible,S., *"Fractional Programming: Applications and Algorithms"*, European Journal of Operations Research, Vol.7, 1981, pp.111-120.

[166] Schaible,S., *"Fractional Programming with Sums of Ratios"*, Scala and Vector Optimization in Economic and Financial Problems, Procedding of the Italian Workshop, (eds.) Castagnoli,E., Giorgi,E., Stampato da Elioprint, 1996, pp.163-175.

[167] Scott,C.H., Jefferson, T.R., *"Fractional Programming Duality via Geometric Programming Duality"*, Journal of Australian Mathematical Society, Sseies B, Vol.21, 1980, pp.398-401.

[168] Seshan,C.R., *"On duality in Linear Fractional Programming"*, Proc. Indian Acad. Sci., Sect. A. Math.Sci., Vol.89, 1980, pp. 35-42.

[169] Seshan,C.R., Tikekar,V.G., *"Algorithms for Integer Fractional Programming"*, Journal of Indian Institute of Science, Vol.62, No.2, 1980, pp.9-16.

[170] Sharma,I.C., Swarup,K., *"On Duality in Linear Fractional Functionals Programming"*, Zeitschrift für Operations Research, Vol.16, 1972, pp.91-100.

[171] Shigeno,M., Saruwatari,Y., Matsui,T., *"An Algorithm for Fractional Assignment Problems"*, DAMATH: Discerete Applied Mathematics and Combinatorial Operations Research and Computer Science, Vol.56, 1995.

[172] Shor,N.Z., Solomon,D.I., *"Decomposition Methods in Linear Fractional Programming"*, Chişinău, "Ştiinţa", 1989.

[173] Skeel,R.D., *"Scaling for Stability in Gaussian Elimination"*, J. Assoc. Comput. Mach., Vol.26, pp.494-526, 1979.

[174] Sniedovich,M., *"Fractional Programming Revised"*, European Journal of Operations Research, Vol.33, pp.334-341, 1988.

[175] Stancu-Minasian,I.M., *"Fractional Programming: Theory, Methods and Applications"*, Kluwer Academic Publishers, 1997.

[176] Suhl,U.H., Suhl,L.M., *"Computing Sparse LU Factorization for Large-scale Linear Programming Bases"*, ORSA Journal of Computing, Vol.2, 1990, pp.325-335.

[177] Swarup,K., *"Some Aspects of Duality for Linear Fractional Functional Programming"*, Zeitschrift für Angewandte Mathematik und Mechanik, Vol.47, No.3, 1967, pp.204-205.

[178] Swarup,K., *"Duality in Fractional Programming"*, Unternehmensforshung, Vol.12, No.2, 1968, pp.106-112.

[179] Taha,H.A., *"Integer Programming. Theory, Applications, and Computations"*, Academic Press, 1975.

[180] Taha,H.A., *"Operations Research, An Introduction"*, Collier MacMillan, New York, 1976.

[181] Terlaky,T., *"A New, Finite Criss-Cross Method for Solving Linear Programming Problems"*, Alkalmazott Matematikai Lapok, Vol.10, 1984, pp.289-296. (in Hungarian).

[182] Terlaky,T., *"A Convergent Criss-Cross Method"*, Math. Oper. und. Statist. Ser. Optim. Vol.16, No.5, 1985, pp.683-690.

[183] Terlaky,T., Zhang,S., *"Pivot Rules for Linear Programming: a Survey on Recent Theoretical Developments"*, Annals of Operations Research, Vol.46, 1993, pp.203-233.

[184] Thiriez,H., *"GULP (version 4.1)"*, in European Journal of Operational Research, Vol.39, 1989, pp.345-346. North-Holland.

[185] Thiriez,H., *"GULF (version 2.2)"*, in European Journal of Operational Research, Vol.67, 1993, pp.295-296. North-Holland.

[186] Vanderbei,R.J., *"Linear Programming. Foundations and Extensions"*, International Series in Operations Research & Management Science, Kluwer Academic Publishers, 1996.

[187] Verma,V., Bakshi,H.C., Puri,M.C., *"Ranking in Integer Linear Fractional Programming Problems"*, Zeitschrift für Operations Research, Vol.34, 1990, pp.325-334.

[188] Wilkinson,J.H., Reinsch,C., *"Linear Algebra"*, Vol.2 of *"Handbook for Automatic Computaion"*, New York, Springer-Verlag, 1971.

[189] Williams,H.P., *"Model Building in Mathematical Programming"*, John Wiley & Sons, A Wiley - Interscience Publication, 1985.

[190] Winston,W.L., *"Indtroduction to Mathematical Programming. Applications & Algorithms"* , PWS-Kent Publishing Company, Boston, 1991.

[191] Wolfe,P., Cutler,L., *"Experiments in Linear Programming"*, In Recent Advances in Mathematical Programming, Ed. Graves,R.L., and Wolfe,P., McGraw-Hill, New York, 1963, pp. 177-200.

[192] Zolkiewski,Z., *"A Multicriteria Linear Programming Model with Linear Fractional Objective Functions"*, Ph.D. Thesis, Central School of Planning and Statistics, Warsaw, Poland, 1983. (in Polish)

[193] Yudin,D.B., Gol'stein,E.G., *"Linear programming (Theory, Methods and Applications)"*, Nauka, Moscow, 1969. (in Russian)