EÖTVÖS LORÁND TUDOMÁNYEGYETEM
TERMÉSZETTUDOMÁNYI KAR
Operációkutatási Tanszék

Egészértékű programozási feladatok megoldása

Bsc Szakdolgozat

Témavezető:
Király Tamás
egyetemi docens

Készítette:
Szöllősi Szilvia
Matematika Bsc
Elemző szakirány

Budapest, 2017

Tartalomjegyzék

1. Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek Király Tamásnak, aki szakértelmével, idejével és hasznos tanácsaival hozzájárult a szakdolgozatom elkészüléséhez. Továbbá köszönettel tartozom középiskolai matematika tanáromnak, Dudás Ljudmillának, akinek hála felfedeztem a matematikában rejlő szépséget. Végül hálával tartozom élettársamnak és szüleimnek, akiktől rengeteg támogatást kaptam az egyetemi éveim alatt.

2. Bevezetés

Szakdolgozatom célja az egészértékű programozás gyakorlati alkalmazásainak bemutatása. Optimalizálási feladatok megoldása az élet legtöbb területén megjelenik, hiszen minden esetben a vezetők, vállalkozók célja az, hogy a bevételek maximálisak, a költségek pedig minimálisak legyenek. A szakdolgozatom elején összefoglalom azokat az előismereteket, amelyek szükségesek az optimalizálási feladatok megoldása során. Az elméleti összefoglaló után pedig különböző, gyakran megjelenő problémákra adok megoldást, felírva lineáris programozási feladatként. A feladatok optimális megoldását a Gusek program segítségével határoztam meg. A szakdolgozatom készítése során megtapasztalhattam, hogy különböző matematikai szoftverek ismerete nélkülözhetetlen, mert olyan problémákra is találhatunk megoldást, amelyet kézzel nehezen vagy nem lehetne meghatározni, valamint a megoldásra szánt idő a töredékére csökkenthető.

3. Szükséges előismeretek

Források: [6] [3] [1]

3.1. Lineáris algebrai egyenletrendszerek (LAER)

1. Definíció (Általános alak). *Adott $A \in \mathbb{R}^{m \times n}$ mátrix, $b \in \mathbb{R}^m$ vektor. Keressük azt az $x \in \mathbb{R}^n$ vektort, mely kielégíti az alábbi feltételt:*

$$Ax = b$$

2. Definíció (Mátrix rangja). *Egy A mátrix rangja r, ha A oszlop/sor-vektorai között található r db lineárisan független, de $r+1$ nem.*

1. Tétel (Kronecker-Capelli). *Az $Ax = b$ lineáris egyenletrendszer akkor és csak akkor megoldható, ha az A együtthatómátrix, és a b vektorral kibővített együttható mátrix rangja megegyezik: $r(A) = r(A \mid b)$. Ha $r(A) < n$ akkor végtelen sok megoldás van, ha $r(A) = n$ akkor egyértelmű a megoldás.*

3.2. LP feladat

A lineáris programozási feladat olyan optimalizálási feladat, ahol a döntési változók egy lineáris függvényét akarjuk maximalizálni bizonyos korlátozó feltételek mellett.
Alakja:

$$\begin{gathered} Ax \leq b \\ x \geq 0 \\ max\ cx \\ x \in \mathbb{R}^n \end{gathered}$$

ahol $A \in \mathbb{R}^{m \times n}$ együtthatómátrix és $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ ismert vektorok.
Egyenletrendszerek megoldását sokkal könnyebb meghatározni, mint egyenlőtlenségrendszerekét, ezért az LP feladat megoldásainak meghatározásához a kanonikus alakot használjuk:

$$\begin{gathered} Ax = b \\ x \geq 0 \\ max\ cx \\ x \in \mathbb{R}^n \end{gathered}$$

Minden LP feladat felírható kanonikus alakban ún. eltérésváltozók bevezetésével. Legyen z az eltérésváltozók vektora, és tekintsük az alábbi feladatot:

$$Ax + Iz = b$$
$$x \geq 0$$
$$max\ cx$$
$$x \in \mathbb{R}^n$$

Ez a feladat már kanonikus alakú, és optimális megoldásai megegyeznek az eredeti feladat optimális megoldásaival.

3. Definíció (Duális feladat)**.** *Primál feladat:*

$$Ax = b$$
$$x \geq 0$$
$$max\ cx$$
$$x \in \mathbb{R}^n$$

Ahol A teljes sorrangú mátrix.
Primál feladathoz tartozó duál feladat:

$$yA \geq c$$
$$min\ yb$$
$$y \in \mathbb{R}^n$$

Az egymáshoz tartozó primál és duál feladatban a célfüggvényértékek megegyeznek. A gyakorlatban a primál feladatban a bevételeket maximalizáljuk, míg a duál feladatban a költségeket minimalizáljuk. A korlátozó feltételek általában az elérhető erőforrásokra vonatkoznak.

Egészértékű LP feladat: A gyakorlatban sokszor olyan problémák merülnek fel, ahol a keresett döntési változók csak egész értéket vehetnek fel, mivel bizonyos mértékegységeknek nem lehet törtrészét meghatározni (Pl.:személyzet, darabszám).

Alakja:

$$Ax = b$$
$$x \geq 0$$
$$max\ cx$$
$$x \in \mathbb{Z}^n$$

ahol $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$.
Speciális eset: bináris programozási feladat, ahol a változó értékeire az alábbi feltétel teljesül: $x \in \{0,1\}^n$

4. Definíció (megoldás). *Azt mondjuk, hogy $x \in \mathbb{R}^n$ vektor megoldása a kanonikus alakú LP feladatnak, ha $Ax = b$ feltétel teljesül.*

5. Definíció (megengedett megoldás). *Azt mondjuk, hogy $x \in \mathbb{R}^n$ vektor megengedett megoldása a kanonikus alakú LP feladatnak, ha $Ax = b$ és $x \geq 0$ feltételek teljesülnek.*

6. Definíció (optimális megoldás). *Azt mondjuk, hogy az $x \in \mathbb{R}^m$ vektor optimális megoldása a kanonikus alakú LP feladatnak, ha x megengedett megoldás és tetszőleges $\tilde{x}$ megoldás esetén $cx \geq c\tilde{x}$*

7. Definíció (Bázis). *Bázisnak nevezünk egy olyan $\mathcal{B}$ leképezést a sorok halmazából az oszlopok halmazába, amelyre $a_{\mathcal{B}_1} \dots a_{\mathcal{B}_m}$ oszlopvektorok lineárisan függetlenek.*

1. Megjegyzés. *Bázisnak rendezett halmazt tekintünk.*

8. Definíció (Bázishoz tartozó bázismátrix). $\begin{bmatrix} a_{\mathcal{B}_1} & \dots & a_{\mathcal{B}_m} \end{bmatrix} := B$

Jelölés: $x_{\mathcal{B}}$ jelölje az x vektor bázishoz tartozó részét, $c_{\mathcal{B}}$ pedig a c vektor bázishoz tartozó részét.

9. Definíció ($\mathcal{B}$ bázishoz tartozó megoldás). *Ha $x \in \mathbb{R}^n$ vektorra teljesül, hogy a bázison kívüli komponensei 0-ák, és $Bx_{\mathcal{B}} = b$ akkor x-et a $\mathcal{B}$ bázishoz tartozó bázismegoldásnak nevezzük.*

Transzformálás : $Ax = b$ egyenlet mindkét oldalát megszorozzuk B^{-1}-gyel. Ekkor a bázismátrix helyére az egységmátrix kerül. $B^{-1}A = \begin{bmatrix} \dots & I & \dots \end{bmatrix} := \bar{A}$, $B^{-1} := \bar{b}$.
A c vektor pedig változatlan marad a transzformáció elvégzése után.

3.3. Szimplex módszer

Kezdeti megoldás:

$$0\bar{a}_j + \sum_{i=1}^{m} x_{\mathcal{B}_i}\bar{a}_{\mathcal{B}_i} = \bar{b}$$

Mivel $a_{\mathcal{B}_i} = e_i$, ezért a kezdeti megoldás $\bar{x}_{\mathcal{B}_i} = \bar{b}_i$. Növeljük $\bar{x}_j$-t 0-ról ϑ-ra.

$$\vartheta\bar{a}_j + \sum_{i=1}^{m}(\bar{b}_i - \vartheta\bar{a}_{ij})e_i = b$$

Célfüggvény változás:

$$\vartheta c_j - \sum_{i=1}^{m} \vartheta c_{\mathcal{B}_i}\bar{a}_{ij} = \vartheta c_j - \vartheta c_{\mathcal{B}}\bar{a}_j$$

Tehát akkor érdemes elvégezni a lépést, ha $c_j - c_{\mathcal{B}}\bar{a}_j > 0$

10. Definíció (Redukált költség). *A $\mathcal{B}$ bázishoz tartozó redukált költség : $\bar{c} = c_{\mathcal{B}}\bar{A} - c$ (ami a báziskoordinátáiban 0). A j. oszlopvektor javító oszlop, ha $c_{\mathcal{B}}\bar{a}_j < c_j$.*

Ha olyan állapotba érünk, ahol nincs javító oszlopvektor, akkor az eljárás leáll, és optimális megoldást találunk.
Ha a javítóoszlopban nincs > 0 komponens, akkor az eljárás megáll, és nincs optimális megoldás, mert a célfüggvény nem korlátos.

3.4. Vágási módszer

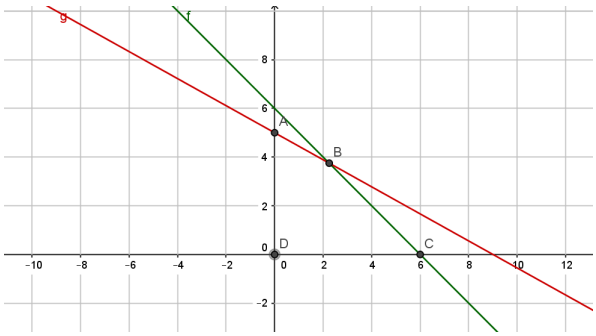
A vágási módszer lényege, hogy az LP feladat megoldását felhasználva meghatározzuk a feladat egészértékű megoldásait. Szimplex módszerrel megoldjuk az LP feladatot, figyelmen kívűl hagyva az egészértékűségi feltételt. Ha a megoldás egész, akkor a feladat készen van, ellenkező esetben pedig új feltételek bevezetésével kikényszerítjük az egészértékűséget. A módszer részletes ismertetéséhez az alábbi példafeladatot használom fel:

$$x_1 + x_2 \leq 6$$
$$5x_1 + 9x_2 \leq 45$$
$$x_1, x_2 \geq 0$$
$$z = max\ 5x_1 + 8x_2$$

Első lépésként alakítsuk át a feladatot kanononikus alakúvá s_1 és s_2 eltérésváltozók bevezetésével:

$$x_1 + x_2 + s_1 = 6$$
$$5x_1 + 9x_2 + s_2 = 45$$
$$x_1, x_2, s_1, s_2 \geq 0$$
$$z = max\ 5x_1 + 8x_2$$

Szemléletes megoldás: Az egyenlőtlenségek által határolt poliéder minden pontja megengedett megoldás lesz, és az optimális megoldás a poliéder valamelyik csúcspontjának felel meg.



Szimplex táblázatot felírva a bekarikázott elemeket használva pivot elemként Gauss-eleminációs műveleteket végeztem.

	x_1	x_2	s_1	s_2	
s_1	(1)	1	1	0	6
s_2	5	9	0	1	45
	-5	-8	0	0	0

	s_1	x_2	s_1	s_2	
x_1	1	1	1	0	6
s_2	0	(4)	-5	1	15
	0	-3	5	0	30

	s_1	x_2	s_1	s_2	
x_1	1	1	1	0	6
s_2	0	(1)	$-\frac{5}{4}$	$\frac{1}{4}$	$\frac{15}{4}$
	0	-3	5	0	30

	s_1	s_2	s_1	s_2	
x_1	1	0	$\frac{9}{4}$	$-\frac{1}{4}$	$\frac{9}{4}$
x_2	0	1	$-\frac{5}{4}$	$\frac{1}{4}$	$\frac{15}{4}$
	0	0	$\frac{5}{4}$	$\frac{3}{4}$	$\frac{165}{4}$

Az utolsó táblázatból az alábbi feltételek olvashatók le:

$$x_1 + \tfrac{9}{4}s_1 - \tfrac{1}{4}s_2 = \tfrac{9}{4}$$
$$x_2 - \tfrac{5}{4}s_1 + \tfrac{1}{4}s_2 = \tfrac{15}{4}$$
$$-\tfrac{5}{4}s_1 - \tfrac{3}{4}s_2 = -\tfrac{165}{4}$$
$$x_1, x_2, s_1, s_2 \geq 0$$

Az egyenlőtlenségekben az együtthatók alsó egészrészét bal oldalra, a törtrészét pedig jobb oldalra rendezve az alábbi alakot kapjuk:

$$x_1 + 2s_1 - s_2 - 2 = \tfrac{1}{4} - \tfrac{1}{4}s_1 - \tfrac{3}{4}s_2$$
$$x_2 - 2s_1 - 3 = \tfrac{3}{4} - \tfrac{3}{4}s_1 - \tfrac{1}{4}s_2$$
$$-2s_1 - s_2 + 42 = \tfrac{3}{4} - \tfrac{3}{4}s_1 - \tfrac{1}{4}s_2$$
$$x_1, x_2, s_1, s_2 \geq 0$$

Az egyenlet jobb oldala egészértékű kifejezés (mivel a bal oldalon is egészértékű kifejezés szerepel), valamint s_1 és s_2, (amelyekről feltettük, hogy nemnegatívak) negatív együtthatókkal jelennek meg. Ezért a jobb oldalon szereplő kifejezésekre az alábbi egyenlőtlenségek írhatók fel:

$$\frac{1}{4} - \frac{1}{4}s_1 - \frac{3}{4}s_2 \leq 0$$
$$\frac{3}{4} - \frac{3}{4}s_1 - \frac{1}{4}s_2 \leq 0$$
$$\frac{3}{4} - \frac{3}{4}s_1 - \frac{1}{4}s_2 \leq 0$$

A kanonikus alakú feladatból s_1 és s_2 kifejezhető x_1 és x_2 segítségével:

$$s_1 = 6 - x_1 - x_2$$
$$s_2 = 45 - 5x_1 - 9x_2$$

Ha ezt visszahelyettesítjük az előbb kapott egyenlőtlenségekbe, akkor az alábbi alakot kapjuk:

$$x_1 + x_2 \leq 6$$
$$5x_1 + 9x_2 \leq 45$$

Ezen feltételekkel és az eredeti célfügvénnyel : $z = 5x_1 + 8x_2$ meghatározott optimalizálási feladat megoldása egészértékű lesz, és megegyezik az eredeti feladat optimális egészértékű megoldásával.
Megoldás: $x_1 = 0, x_2 = 5, z = 40$
A módszer alkalmazása azonban nem minden esetben ilyen egyszerű, mint a fenti példa-feladatban. A módszer iterációs eljárás, azaz a vágási lépéseket addig kell ismételni, míg egészértékű megoldáshoz nem jutunk.

3.5. Logikai feltételek megadása

Gyakran előfordul, hogy a feladat megoldását korlátozó feltételek között logikai kapcsolat van. Ebben a részben megmutatom, hogy ezek a kapcsolatok hogyan építhetők bele a lineáris programozási modellbe.

1. **Döntési változók egymáshoz való viszonya:**
   Legyenek $x_1 \ldots x_n$ bináris döntésváltozók, melynek értékeire:
   $$x_i := \begin{cases} 1 & \text{ha az i. lehetőség megvalósul} \\ 0 & \text{különben} \end{cases}$$
   - $1 \ldots k$ lehetőségekből maximum j valósulhat meg: $\sum_{i=1}^{k} x_i \leq j$
   - Ha az i. lehetőség megvalósul, akkor a j-ediket is meg kell valósítanunk: $x_i - x_j \leq 0$ Ekkor ha $x_i = 1$ akkor $x_j = 1$, ellenkező esetben pedig $x_i = 0$ akkor $x_j = 0$ vagy 1.
   - Ha az i. lehetőség megvalósul, akkor a j. lehetőséget már nem valósíthatjuk meg: $x_i + x_j \geq 1$ Ha $x_i = 1$ akkor $x_j = 0$, ellenkező esetben pedig $x_i = 0$ akkor $x_j = 0$ vagy 1.

2. **Alternatív feltételek:** Legyenek $x_1 \dots x_n$ a feladat megoldását meghatározó változók, és az alábbi két feltétel közül legalább az egyiknek teljesülnie kell:

$$f_1(x_1 \dots x_n) \leq b_1$$

$$f_2(x_1 \dots x_n) \leq b_2$$

ahol f_1 és f_2 lineáris függvények. Ebben az esetben bevezetünk egy y bináris változót, valamint választunk B_1 és B_2 elég nagy konstansokat úgy, hogy az alábbi összefüggések érvényesek legyenek.

$$\begin{gathered} f_1(x_1 \dots x_n) - B_1 y \leq b_1 \\ f_2(x_1 \dots x_n) - B_1(1-y) \leq b_2 \\ y = 0 \text{ vagy } 1 \end{gathered}$$

Ebben az esetben ha $y = 1$ akkor a 2. feltétel teljesül, ellenkező esetben pedig az 1.

3. **k feltétel:** Adottak az alábbi feltételek:

$$f_j(x_1 \dots x_n) \leq b_j \; j = 1 \dots p$$

Ha a megadott j feltételből legalább k-nak teljesülnie kell, akkor ezt az alábbi összefüggésekkel biztosíthatjuk:

$f_j(x_1 \dots x_n) - B_j(1 - y_j) \leq b_j$ ahol a B_j-k megfelelően nagy konstansok.

$$\sum_{i=1}^{p} y_j \geq k$$

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

4. **Komponensekre osztott feltételek:** Adott 3 db feltételhalmaz. A halmazon belüli feltételeknek egyszerre kell teljesülniük, valamit legalább 2 halmaz feltételeinek teljesülnie kell. Ez a probléma az alábbi feltételekkel fogalmazhatók meg.

$$
\begin{cases}
f_1(x_1 \dots x_n) - B_1 y_1 \leq b_1 \\
f_2(x_1 \dots x_n) - B_1 y_1 \leq b_2
\end{cases}
\begin{cases}
f_3(x_1 \dots x_n) - B_3 y_2 \leq b_3 \\
f_4(x_1 \dots x_n) - B_4 y_2 \leq b_4 \\
f_5(x_1 \dots x_n) - B_5 y_2 \leq b_5
\end{cases}
\begin{cases}
f_6(x_1 \dots x_n) - B_6 y_3 \leq b_1 \\
f_7(x_1 \dots x_n) - B_7 y_3 \leq b_7
\end{cases}
$$

$$
\begin{gathered}
y_1 + y_2 + y_3 \geq 2 \\
x_i \geq 0 \; i = 1 \dots n \\
y_1, y_2, y_3 \text{ bináris változók.}
\end{gathered}
$$

4. Feladatok megoldása

Források: [1] [2] [5] [4]
Ebben a fejezetben a gyakorlatban sokszor előforduló feladatok megoldásai szerepelnek. A feladat leírása után ismertetem a feladat megoldásához szükséges paramétereket, a megoldást meghatározó változót (változókat abban az esetben ha több kérdésre akarunk választ adni, vagy pedig segédváltozóra van szükség bizonyos feltételek teljesüléséhez). Ezek után a korlátozó feltételeket adom meg, és végül a feladatot megoldó programkód szerepel az általa meghatározott optimális megoldással együtt.

4.1. Utazó ügynök probléma

Feladat leírása: Adott egy ügynök, aki az otthonából indulva 6 db magyar várost szeretne meglátogatni és visszatérni az otthonába úgy, hogy minden városban csak egyszer jár. Célunk megadni a városok meglátogatási sorrendjét úgy, hogy az utazási költségek minimálisak legyenek. A probléma egy $G = (V, E)$ teljes irányított gráffal szemléltethető, melynek csúcsai a városoknak felelnek meg, és minden élen kétoldali irányítást veszünk. Az éleken megfeleltetünk egy költségfüggvényt $c : E \to R^+$ amely az él két végpontjának megfelelő városok közötti utazási költségnek felel meg.

11. Definíció (Hamilton kör)**.** *Egy G gráf körét Hamilton-körnek nevezzük, ha a gráf minden csúsát tartalmazza.*

A feladat megoldása egy irányított Hamilton-kör lesz.
Az I és J tömbök a városoknak felelnek meg.
Ismert paraméterek

- c: c_{ij} = az i. és j. város távolsága km-ben megadva, melynek pontos értékeit [8] alapján határoztam meg.
- varosok: A városok pontos neveit tartalmazza

Változók:

- $x : x_{ij} = \begin{cases} 1 & \text{ha az i. városból a j. városba utazik} \\ 0 & \text{különben} \end{cases}$
- u segédváltozó: ui = az i. város meglátogatásának sorszáma. Ezt a változót a diszjunkt körök elkerülése miatt szükséges bevezetni.

Célfüggvény:

- ossztav: Az ügynök által összesen megtett távolságot akarjuk minimalizálni.

Változót korlátozó feltételek:

- egy_be: Ez a feltétel biztosítja, hogy minden városba csak egyszer fog belépni
- egy_ki: Ez a feltétel biztosítja, hogy minden városból csak egyszer fog kilépni
- u_def: Ez a feltétel határozza meg az u értékét, hiszen ha $x_{ij} = 0$ akkor a sorrendbeli távolságuk legfeljebb 5 lehet, ha pedig $x_{ij} = 1$ akkor a sorrendbeli távolságuk pontosan 1.
- u_min: u minimális értékének beállítása

Megoldás:

```
set I:=1..6;
set J:=1..6;

param c {i in I, j in J};
param varos {i in I} symbolic;

var x {i in I, j in J} binary;
var u {i in I};

minimize ossztav : sum {i in I, j in J} x[i,j]*c[i,j];

egy_be {j in J}: sum {i in I} x[i,j]=1;
egy_ki {i in I}: sum {j in J} x[i,j]=1;
u_def {i in 2..6, j in 2..6}: u[i]-u[j]+1<=(6-1)*(1-x[i,j]);
u_min {i in 2..6}: u[i]>=2;



solve;

printf "\n";
for {i in 1..6, j in 1..6}
{ printf(if(x[i,j]>0) then " %s  %s  : %s \n  " else ""), ↩
   varos[i], varos[j], c[i,j];
}
printf"\n";
printf "Osszesen megtett tavolsag: ";
printf ossztav;

data;
```

```
param c:    1     2     3     4     5     6:=
        1   0     99    221   170   120   102
        2   99    0     130   120   215   198
        3   221   130   0     214   340   265
        4   170   120   214   0     265   202
        5   120   215   340   265   0     115
        6   102   198   322   202   115   0;

param varos:= 1    Budapest
              2    Szolnok
              3    Debrecen
              4    Szeged
              5    Gyor
              6    Siofok;
end;
```

Budapest Gyor : 120
Szolnok Budapest : 99
Debrecen Szolnok : 130
Szeged Debrecen : 214
Gyor Siofok : 115
Siofok Szeged : 202

Osszesen megtett tavolsag: 880

4.2. Logisztikai problémák

4.2.1. Repülési társaság személyzetének ütemezése

Forrás: [7]
Feladat leírása: Adott 7 db repülési útvonal (járat), és 5 db repülési szakasz az Amerika Egyesült Államok nagyobb városain keresztül . Célunk a személyzet elosztása az útvonalakon úgy, hogy minden szakaszon közlekedjen járat, és a személyzet költsége minimális legyen.
Az I tömb elemei a szakaszoknak, a J tömb elemei pedig az útvonalaknak felelnek meg.
Ismert paraméterek:

- a: $a_{ij} := \begin{cases} 1 & \text{ha a j. útvonal tartalmazza az i. szakaszt} \\ 0 & \text{különben} \end{cases}$
- c: $c_j :=$ A j. útvonalon a személyzet költsége.

- jarat: Megadja, hogy az egyes járatok mely városokon át közlekednek.
- szakasz: Két város közötti szakaszokat adja meg.

Változó:

- x: $x_j := \begin{cases} 1 & \text{ha a j. útvonalon indul járat} \\ 0 & \text{különben} \end{cases}$

Célfüggvény:

- osszkoltseg: Az indított járatok személyzeti költségét minimalizálja.

Korlátozó feltétel:

- felt: Biztosítja hogy minden útvonalon csak 1 járatra elegendő személyzet tartozik, valamint minden szakaszt csak 1 járat tartalmaz.

Megoldás:

```
set I;
set J;

param a {i in I, j in J};
param c {j in J};
param jarat {j in J} symbolic;
param szakasz {i in I} symbolic;

var x{j in J} binary;

minimize osszkoltseg: sum{ j in J} x[j]*c[j];

felt {i in I}: sum {j in J} a[i,j]*x[j]=1;

solve;
printf"\n";
printf "Inditott jaratok:";
printf"\n";
for {j in J}
{printf (if(x[j]=1) then "%s \n" else ""), jarat[j];
}
printf"\n";
printf"Koltseg: ";
printf osszkoltseg;
printf"\n";
```

```
data;

set I:=1 2 3 4 5;
set J:=1 2 3 4 5 6 7;

param a:      1   2   3   4   5   6   7:=

      1       0   0   0   0   1   1   1
      2       1   1   0   0   0   0   0
      3       0   0   1   1   0   0   0
      4       1   0   0   0   0   0   1
      5       0   1   0   1   0   1   0;

param jarat:=
1    San_Francisco___Denver___New_York___San_Francisco
2    San_Francisco___Denver___Las_Vegas___San_Francisco
3    Pantnagar___Las_Vegas___Pantnagar
4    Pantnagar___Las_Vegas___San_Francisco___Pantnagar
5    New_York___Boston___San_Francisco___New_York
6    San_Francisco___New_York___Boston___Las_Vegas___San_Francisco
7    New_York___Boston___Denver___New_York;

param szakasz:=
1    New_York___Boston
2    San_Francisco___Denver
3    Las_Vegas___Pantnagar
4    Denver___New_York
5    Las_Vegas___San_Francisco;

param c:=
1   560
2   335
3   420
4   470
5   545
6   660
7   490;

end;
```

Inditott jaratok:
San_Francisco Denver Las_Vegas San_Francisco
Pantnagar Las_Vegas Pantnagar
New_York Boston Denver New_York

Koltseg: 1245

Alternatív feltétel:

$$\sum_{j=1}^{7} a_{ij}x_j \geq 1, i = 1 \dots 5$$

Ezek a feltételek megengedik, hogy bizonyos szakaszokon a személyzet mint utas tartózkodjon. Erre akkor lehet szükség, ha a személyzetnek olyan indulási állomással rendelkező útvonalon kell dolgoznia, amelyet csak repülés útján tud megközelíteni.

4.2.2. Szállítási feladat

Feladat leírása: Adott 6 db feladó állomás és 5 db felvevő állomás. Üres konténereket akarunk szállítani a feladó állomásokról a felvevő állomásokra, úgy hogy a felvevő állomások konténer kereslete ki legyen elégítve és a szállítási költség minimális legyen.
Az I tömb elemei a feladó állomások, a J tömb elemei pedig a felvevő állomások. A paraméterek értékeit ezen tömbök elemeihez rendeljük hozzá.
Ismert paraméterek:

- a: a feladó állomásokon található kontérek száma
- b: a felvevő állomások konténer iránti kereslete
- d: a feladó és felvevő állomások közötti távolság km-ben
- f: az 1 km-re jutó szállítási költség
- c: a megadott paraméterekből kiszámított érték, mely a feladó- és felvevő állomás közötti szállítási költséget adja meg

Változó:

- x: x_{ij} az i. városból j. városba szállítandó konténerek száma

Célfüggvény:

- koltseg: a szállítási költség minimalizálása

Korlátozó feltételek:

- felad: Meg van határozva, hogy a feladó állomásokon mennyi konténer található, ennél többet nem szállíthatunk.
- atvesz: Meg van határozva a felvevő állomások konténer iránti kereslete, ennél kevesebbet nem szállíthatunk.

Megoldás:

```
set I;
set J;

param a{i in I};
param b{j in J};
param d{i in I, j in J};
param f;
param c{i in I, j in J} := f * d[i,j];

var x {i in I, j in J}>=0 integer;

minimize koltseg: sum{i in I, j in J} c[i,j] * x[i,j];

felad {i in I}: sum{j in J} x[i,j] <=a[i];
atvesz {j in J}: sum{i in I} x[i,j] >= b[j];
solve;

printf "\n";

for {i in I, j in J}
{printf (if (x[i,j] > 0) then " %s -> %s : %s \n" else ""), i, ↩
   j, x[i,j];
}
printf"\n";
printf "Koltseg";
printf koltseg;
print "\n";
data;

set I := Verona Perguia Roma Pescara Taranto Lamezia;
set J := Genoa Velence Ancona Naples Bari;
```

```
param a := Verona     10
           Perguia    12
           Roma       20
           Pescara    24
           Taranto    18
           Lamezia    40;

param b := Genoa   20
           Velence 15
           Ancona  25
           Naples  33
           Bari    21;

param d :       Genoa Velence Ancona Naples Bari :=
      Verona    290   115     355    715    810
      Perguia   380   340     165    380    610
      Roma      505   530     285    220    450
      Pescara   655   450     155    240    315
      Taranto   1010  840     550    305    95
      Lamezia   1072  1097    747    372    333;

param f:=30;

end;
```

Verona → Velence : 10
Perguia → Genoa : 7
Perguia → Velence : 5
Roma → Genoa : 13
Roma → Ancona : 1
Roma → Naples : 6
Pescara → Ancona : 24
Taranto → Bari : 18
Lamezia → Naples : 27
Lamezia → Bari : 3

Koltseg: 90459

4.3. Termelési feladatok

4.3.1. Carreland

Feladat leírása: Egy képzeletbeli Carreland nevű állam acélt, motoros gépeket, elektromos alkatrészeket és műanyagot termel. Az állam pénzneme Klunz. Célunk meghatározni, hogy mennyit termeljünk a különböző termékekből, hogy az állam exportból származó bevétele maximális legyen. A termékek előállításához szükség van a többi termék felhasználására, az előállítási költség kifizetésére és munkaerő felhasználására.
Az I tömb elemei az előállítható termékek.
Ismert paraméterek:

- maxmunkas: az állam elérhető munkaereje fő/év mértékegységben
- szuk_munkas: a termékek előállításához szükséges munkaerőt fő/év mértékegységben
- maxtermelheto: Az egyes termékekből maximálisan előállítható mennyiség
- eladar: a termékek eladási ára az exportálás során
- szuk_termekek: az egyes termékek előállítása során hány egységet kell felhasználni a többi termékből
- elo_ktg: a termékek előállítási költsége

Változók:

- x: x_i =az i. termékből gyártott mennyiség
- y: y_i =az i. termékből exportált mennyiség

Célfüggvény:

- bevetel: A bevétel maximalizálása, melynek értékét növeli az exportált termékek eladása, és csökkenti a termékek előállítása során felmerülő költségek.

Korlátozó feltételek:

- maxeloallithato: Nem termelhetünk többet, mint amennyit az állam maximálisan előállíthat
- max_munkaero: Nem használhat fel az állam több munkaerőt, mint amennyivel rendelkezik
- egyensuly: Az exportált és a termelés során felhasznált termékek összege megegyezik az előállított termékek mennyiségével.

Megoldás:

```
set I;

param maxmunkas;
param szuk_munkas {i in I};
param maxtermelheto {i in I};
param eladar{i in I};
param szuk_termekek {i in I, j in I};
param elo_ktg {i in I};

var  x {i in I}>=0 integer;
var y {i in I}>=0 integer;

maximize bevetel : sum{i in I} (y[i]*eladar[i]) - sum {i in I} ↩
   (elo_ktg[i]*x[i]);

maxeloallithato {i in I}: x[i]<=maxtermelheto[i];
max_munkaero: sum {i in I} szuk_munkas[i]*x[i]<=maxmunkas;
egyensuly {i in I}: y[i]+ sum {j in I} ↩
   szuk_termekek[i,j]*x[j]=x[i];

solve;

printf "\n";
printf "Termelt";
printf "\n";

for {i in I}
{printf (if (x[i]>=0) then "%s : %s \n" else ""), i, x[i];}
printf "\n";
printf "Export";
printf "\n";
for {i in I}
{printf (if (y[i]>=0) then "%s : %s \n" else ""), i, y[i];}
printf "\n";
printf "\n"
printf "Bevetel: "
printf bevetel;
printf"\n";
data;
```

```
set I:=acel mot_gepek elek_alkat muanyag;

param maxmunkas:=830000;

param szuk_munkas:= acel          0.5
                    mot_gepek     1
                    elek_alkat    0.5
                    muanyag       2;

param maxtermelheto:=acel         90000000000
                     mot_gepek    650000
                     elek_alkat   90000000000
                     muanyag      60000;

param eladar:=acel          500
              mot_gepek     1500
              elek_alkat    300
              muanyag       1200;

param szuk_termekek:
             acel      mot_gepek    elek_alkat   muanyag:=
acel         0         0.02         0            0.01
mot_gepek    0.8       0            0.15         0.11
elek_alkat   0.01      0.01         0            0.05
muanyag      0.2       0.03         0.05         0;

param elo_ktg:=acel             250
               mot_gepek        300
               elek_alkat       50
               muanyag          300;

end;
```

Termelt:
acel : 13600
mot_gepek : 650000
elek_alkat : 9640
muanyag : 60000
Export:
acel : 0
mot_gepek : 631074
elek_alkat : 4
muanyag : 37298
Bevetel: 774487800

4.3.2. Vegyes termelés

Feladat leírása: Egy cég 3 féle terméket állít elő, melyeket A1, A2 és A3-mal jelölünk. A gyártósor 22 napig üzemel. Célunk meghatározni, hogy melyik termékből mennyit termeljünk úgy, hogy a bevétel maximális legyen.
Az I tömb elemei az előállítható termékek.
Ismert paraméterek:

- maxker: a termékek iránti maximális keresletet 100 kg egységben
- koltseg: a termékek előállításához szükséges költség $ \100 kg mértékegységben
- eladar: a termékek eladási ára $ /100 kg mértékegységben
- kvota: a gyártósor mennyi terméket tudna termelni abban az esetben, ha egy napig csak az egyik termék gyártásával foglalkozna
- napok: a gyártósor működési ideje napokban megadva

Változó:

- x: x_i =az i. termékből gyártott mennyiség

Célfüggvény:

- bevetel: A bevétel maximalizálása, melyet növel az eladott termékek ára, és csökkent a termeléshez szükséges előállítási költség.

Korlátozó feltételek:

- maxtermelheto: Nem állíthatunk elő több terméket, mint amennyi a maximális kereslete.
- napitermelehto: Nem állíthatunk elő több terméket, mint amennyi a gyártósor kapacitása.

Megoldás:

```
set I;

param maxker{i in I};
param koltseg{i in I};
param eladar{i in I};
param kvota{i in I};
param napok;

var x{i in I}>=0 integer;

maximize bevetel : sum {i in I} x[i]*(eladar[i]-koltseg[i]);

maxtermelheto {i in I}: x[i]<=maxker[i];
napitermelheto : sum{i in I} x[i]/kvota[i]<=napok;

solve;

printf "\n"

for {i in I}
{printf (if (x[i]>=0) then "%s:  %d  \n" else ""), i, x[i];
}
printf "\n";
printf "Bevetel: ";
printf bevetel;
printf"\n";
data;

set I:= A1 A2 A3;

param maxker:= A1 5300
               A2 4500
               A3 5400;
```

```
param koltseg:= A1 73.3
                A2 52.9
                A3 65.4;

param eladar:= A1 124
               A2 109
               A3 115;

param kvota:= A1 500
              A2 450
              A3 550;

param napok:= 22;

end;
```

A1: 5300
A2: 712
A3: 5400
Bevetel: 576483

Kiegészítő paraméterek megadása

- akt_ktg: a termékek gyártásának beindításához szükséges aktiválási költség
- min_term: ha a termék gyártása beindul, akkor meg van határozva a minimálisan termelhető mennyiség

Ezen feltételek megváltoztathatják a feladat megoldását, mert előfordulhat, hogy bizonyos termékek gyártását nem éri meg elindítani az aktiválási költség kifizetése miatt.

Új változó bevezetése:

- $y_i := \begin{cases} 1 & \text{az i. termék gyártását beindítjuk} \\ 0 & \text{különben} \end{cases}$

Célfüggvény: A bevétel maximalizálása, melynek érték növeli a termékek eladása, és csökkenti az aktiválási költség és az előállítási költség kifizetése.

Megoldás:

```
set I;

param maxker{i in I};
param koltseg{i in I};
param eladar{i in I};
param kvota{i in I};
param napok;
param akt_ktg{i in I};
param min_term{i in I};

var x{i in I}>=0 integer;
var y {i in I}>=0 binary;

maximize bevetel : sum {i in I} ↩
   (x[i]*(eladar[i]-koltseg[i])-akt_ktg[i]*y[i]);

maxtermelheto{i in I} : x[i]<=maxker[i];
termelheto_e {i in I}: x[i]<=napok*kvota[i]*y[i];
napitermelheto : sum{i in I} x[i]/kvota[i]<=napok;
mintermelheto {i in I}: x[i]>=min_term[i]*y[i];

solve;

printf "\n";
for {i in I}
{ printf (if (x[i]>=0) then "%s : %d  \n" else ""), i, x[i];
}
printf "\n";
printf "Bevetel: ";
printf bevetel;
printf"\n";
data;

set I:= A1 A2 A3;

param maxker:= A1 5300
               A2 4500
               A3 5400;
```

```
param koltseg:= A1 73.3
                A2 52.9
                A3 65.4;

param eladar:= A1 124
               A2 109
               A3 115;

param kvota:= A1 500
              A2 450
              A3 550;

param akt_ktg:= A1 170000
                A2 150000
                A3 100000;

param napok:= 22;

param min_term:= A1 20
                 A2 20
                 A3 16;

end;
```

A1 : 0
A2 : 4500
A3 : 5400
Bevetel: 270290

4.3.3. Hosszú távú termelés tervezés

Feladat leírása: A korábbi feladathoz hasonlóan egy cég 3 féle terméket termel (A1, A2, A3), és célunk a termelés megtervezése a bevétel maximalizálásával. Ebben az esetben viszont 4 hónapon át folyik a termelés, és lehetőség van a termékek raktárba helyezésére is.
Az I tömb elemi a gyártott termékek, a J tömb pedig egész számok 1-től n-ig, ahol n a hónapok számát jelöli.

Ismert paraméterek

- maxker: minden hónapban változik a termék iránti maximális kereslet
- eladar: a termékek eladási ára
- akt_ktg: minden hónapban kifizetendő aktiválási költség, ha a termék gyártásra kerül az adott hónapban
- ktg: az előállításhoz szükséges költség
- min_term: ha a termék gyártása beindul, akkor meg van határozva a minimálisan termelhető mennyiség
- berlet: a tárhely bérleti díja, melyet minden hónapban ki kell fizetni
- tarhely: a tárhely maximális kapacitása
- kvota: a gyártósor mennyi terméket tudna termelni abban az esetben, ha egy napig csak az egyik termék gyártásával foglalkozna
- honapnev: a kiiratáshoz szükséges paraméter, mely a hónapok neveit tartalmazza

Változók:

- x: x_{ij} = Az i. termékből a j. hónapban termelt mennyiség.
- z: z_{ij} =az i. termékből a j. hónapban félretett termékek száma
- w: w_{ij} =az i. termékből a j. hónapban eladott termékek száma
- y: $y_i := \begin{cases} 1 & \text{az i. termék gyártását beindítjuk} \\ 0 & \text{különben} \end{cases}$

Célfüggvény:

- :bevetel: Értékét növeli az eladott termékek ára, és csökkentik a termelési költségek, aktiválási költségek és a bérleti díjak.

Korlátozó feltételek:

- maxeladhato: nem adhatunk el több terméket, mint amennyi a maximális kereslet
- napi_termelheto: nem termelhetünk többet mint amennyi a gyártósor kapacitása
- maxfelrerakhato: meg van határozva a tárhely kapacitása, ennél többet nem tehetünk félre
- termel_e: az y változó értékét adja meg

- min_termeles: meg van határozva, hogy minimum mennyit kell termelnünk az adott termékből, ha a gyártását beindítjuk
- tarhely_egyensuly: Minden hónapban az eladott és félretett termékek összege meg kell hogy egyezzen a gyártott és a megelőző hónapban a tárhelyben lévő termékek összegével.

Megoldás:

```
set I;
param n integer;
set J := 1..n;

param honapnev {j in J} symbolic;
param eladar {i in I};
param akt_ktg {i in I};
param ktg {i in I};
param kvota {i in I};
param min_term {i in I};
param napok {j in J};
param maxker {i in I, j in J};
param berlet {i in I};
param tarhely;

var x {i in I, j in J}, >= 0 integer;
var y {i in I, j in J} binary;
var w {i in I, j in J}, >=0 integer;
var z {i in I, j in J}, >=0 integer;

maximize bevetel: sum{i in I}
(eladar[i] * sum{j in J} w[i,j] - ktg[i] * sum {j in J} ↩
   x[i,j]-berlet[i] * sum{j in J} z[i,j]-akt_ktg[i]*sum{j in J} ↩
   y[i,j]);

maxeladhato {i in I, j in J}: w[i,j]<=maxker[i,j];
napi_termelheto {j in J}: sum{i in I} x[i,j]/kvota[i]<=napok[j];
maxfelrerakhato {j in J}: sum {i in I} z[i,j]<=tarhely;
termel_e {i in I, j in J}: x[i,j]<=napok[j]*kvota[i]*y[i,j];
min_termeles {i in I, j in J}: x[i,j]>=min_term[i]*y[i,j];
tarhely_egyensuly {i in I, j in 2..n}: ↩
   z[i,j-1]+x[i,j]=z[i,j]+w[i,j];

solve;
```

```
printf "\n";
printf"Termelt";
printf"\n";
for {i in I, j in J}
{printf (if (x[i,j] > 0) then " %s  %s : %s \n" else ""), ↩
   honapnev[j], i, x[i,j];
}

printf "\n";
printf"Raktar";
printf"\n";
for {i in I, j in J}
{printf (if (z[i,j] > 0) then " %s  %s : %s \n" else ""), ↩
   honapnev[j], i, z[i,j];
}
printf "\n";
printf"Eladott";
printf"\n";
for {i in I, j in J}
{printf (if (w[i,j] > 0) then " %s  %s : %s \n" else ""), ↩
   honapnev[j], i, w[i,j]; }

printf "Bevetel:" ;
printf bevetel;
printf"\n";

data;

param n := 4;
set I:= A1 A2 A3;

param honapnev :=   1 Jan
                    2 Feb
                    3 Marc
                    4 Apr;

param maxker:  1       2       3        4  :=
            A1 5300    1200    7400     5300
            A2 4500    5400    6500     7200
            A3 4400    6700    12500    13200;
```

```
param eladar:= A1 124
               A2 109
               A3 115;

param akt_ktg:= A1 150000
                A2 150000
                A3 100000;

param ktg := A1 73.3
             A2 52.9
             A3 65.4;

param kvota := A1 500
               A2 450
               A3 550;

param min_term := A1 20
                  A2 20
                  A3 16;

param napok := 1 23
               2 20
               3 23
               4 22;

param berlet:= A1 3.5
               A2 4
               A3 3;

param tarhely:=800;

end;
```

Termelt
Feb A2 : 3519
Feb A3 : 6699
Marc A3 : 12650
Apr A3 : 12100

Raktar
Jan A1 : 800
Marc A3 : 150

Eladott
Jan A1 : 5300
Feb A1 : 800
Jan A2 : 4500
Feb A2 : 3519
Jan A3 : 4400
Feb A3 : 6699
Marc A3 : 12500
Apr A3 : 12250

Bevetel:3056936.3

4.4. Parkolási probléma

Feladat leírása: Mr. Edmonds egy partit rendez 30 főnek, akik 15 autóval érkeznek, melynek ismertek a hosszai méterben megadva. A vendégek az út két oldalán tudnak parkolni az autójakkal. Mr. Edmonds meg akarja szervezni a vendégek parkolását úgy, hogy az autók által elfoglalt hosszúság minimális legyen.
Az I tömb elemei az autók sorszáma, a J tömb eleme pedig: 1, 2, amely az út két oldalának felel meg.
Ismert paraméterek:

- hossz: Az autók ismert hosszait tartalmazza méterben megadva

Változók:

- x: $x_{ij} := \begin{cases} 1 & \text{az i. autó a j. oldalon áll} \\ 0 & \text{különben} \end{cases}$
- y: y_j =a j. oldalon lévő autók által elfoglalt hosszúság
- Y: $Y = max\ y_i$

Célfüggvény

- max_hossz: Az út oldalain elfoglalt maximális hosszt akarjuk minimalizálni.

Korlátozó feltételek:

- csak_egy_oldal: Ez a feltétel biztosítja, hogy minden autó csak az egyik oldalon fog parkolni
- oldalhossz: Biztosítja, hogy az y_j-k valóban az autók által elfoglalt helyet adják meg.
- oldalhossz_max: Y meghatározásához szükséges feltétel.

Megoldás:

```
set I;
set J;

param hossz{i in I};

var x {i in I, j in J} binary;
var y{j in J};
var Y>=0;

minimize max_hossz: Y;

csak_egy_oldal {i in I}: sum{j in J} (x[i,j])=1;
oldalhossz {j in J}: sum {i in I} (hossz[i]*x[i,j])=y[j];
oldalhossz_max {j in J}: Y>=y[j];

solve;

printf "\n\n\n";
for{i in I, j in J}
{printf (if (x[i, j]>0) then "  %d . auto  %d . oldalon van \n " ↩
   else ""), i, j;
}
printf "\n\n\n";
for{j in J}
{ printf (if (y[j]>0) then "%s. oldalon az autok hossza osszesen ↩
   %s meter \n" else ""), j , y[j];
}
printf "\n\n\n";

data;

set I:=1 2 3 4 5 6 7 8 9 10 11 12 13 14 15;
set J:=1 2;
```

```
param hossz:=
1  4
2  4.5
3  5
4  4.1
5  2.4
6  5.2
7  3.7
8  3.5
9  3.2
10 4.5
11 2.3
12 3.3
13 3.8
14 4.6
15 3;

end;
```

1 . auto 2 . oldalon van
2 . auto 1 . oldalon van
3 . auto 2 . oldalon van
4 . auto 2 . oldalon van
5 . auto 1 . oldalon van
6 . auto 1 . oldalon van
7 . auto 2 . oldalon van
8 . auto 1 . oldalon van
9 . auto 2 . oldalon van
10 . auto 1 . oldalon van
11 . auto 2 . oldalon van
12 . auto 2 . oldalon van
13 . auto 1 . oldalon van
14 . auto 1 . oldalon van
15 . auto 2 . oldalon van

1. oldalon az autok hossza osszesen 28.5 méter
2. oldalon az autok hossza osszesen 28.6 méter

4.5. Örökösödési kérdés

Feladat leírása: Egy gazdag arisztokrata azt a végrendeletet hagyta, hogy vagyontárgyait a két fia között kell elosztani, úgy hogy a vagyontárgyak értékei közötti eltérés minimális legyen.
Az I tömb elemei a vagyontárgyak sorszámait jelölik.
Ismert paraméterek:

- ertek: A vagyontárgyak értékei $-ban megadva
- megnevezes: A vagyontárgyak megnevezései

Változók:

- x: $x_i = \begin{cases} 1 & \text{ha az i. vagyontárgyat az 1. testvér kapja} \\ 0 & \text{különben} \end{cases}$
- y: $y_i = \begin{cases} 1 & \text{ha az i. vagyontárgyat a 2. testvér kapja} \\ 0 & \text{különben} \end{cases}$
- a: Az első testvér által kapott vagyontárgyak összértéke
- b: A második testvér által kapott vagyontárgyak összértéke
- z: A két testvér által kapott vagyontárgyak összértékeinek abszolút eltérése

Célfüggvény:

- kulonbseg: A z változó minimalizálása

Változókat korlátozó feltételek:

- egyensuly: Minden vagyontárgyat pontosan az egyik testvér örökölhet
- a_def: Az a változó értékét adja meg
- b_def: A b változó értékét adja meg
- elso_felt és masodik_felt: Biztosítják h a z változó a két testvér által kapott vagyokontárgyak értékének abszolút eltérését adja meg

Megoldás:

```
set I;
param ertek {i in I};
param megnevezes {i in I} symbolic;

var x {i in I} binary;
```

```
var y {i in I} binary;
var z>=0;
var a>=0;
var b>=0;

minimize kulonbseg: z;

egyensuly {i in I}: x[i] + y[i]=1;
a_def: sum{i in I} ertek[i]*x[i]=a;
b_def: sum{i in I} ertek[i]*y[i]=b;
elso_felt : z>=b-a;
masodik_felt: z>=a-b;

solve;
printf "\n";
printf"1. testver altal orokolt vagyontargyak:";
printf"\n";

for{i in I}
{printf (if(x[i]=1) then "%s, %s \n" else ""), megnevezes[i], ↩
   ertek[i];
}

printf"\n";
printf"2. testver altal orokolt vagyontargyak";
printf"\n";
for{i in I}
{printf (if(y[i]=1) then "%s, %s \n" else ""), megnevezes[i], ↩
   ertek[i];
}
printf"\n";
printf "Elteres";
printf kulonbseg;
printf"\n";

data;

set I:=1 2 3 4 5 6 7 8 9 10 11 12 13 14;
```

```
param ertek:=
        1   25000
        2   5000
        3   20000
        4   40000
        5   12000
        6   12000
        7   12000
        8   3000
        9   3000
        10  3000
        11  10000
        12  15000
        13  10000
        14  13000;

param megnevezes:=   1    Caillebotte_festmeny
                     2    Diocletianus_mellszobra
                     3    Yuan_dinasztia_kinai_vaza
                     4    Porsche
                     5    Gyemant_I.
                     6    Gyemant_II.
                     7    Gyemant_III.
                     8    XV_Lajos_kanapeja
                     9    Jack_Russel_kutya_I
                     10   Jack_Russel_kutya_II
                     11   III_szazadi_szobor
                     12   Vitorlas_hajo
                     13   Harley_Davidson_motorkerekpar
                     14   Cavour_butora;

end;
```

1. testver altal orokolt vagyontargyak:
Caillebotte_festmeny, 25000
Yuan_dinasztia_kinai_vaza, 20000
Porsche, 40000
XV_Lajos_kanapeja, 3000
Jack_Russel_kutya_II, 3000

2. testver altal orokolt vagyontargyak
Diocletianus_mellszobra, 5000
Gyemant_I., 12000
Gyemant_II., 12000
Gyemant_III., 12000
Jack_Russel_kutya_I, 3000
III_szazadi_szobor, 10000
Vitorlas_hajo, 15000
Harley_Davidson_motorkerekpar, 10000
Cavour_butora, 13000

Elteres1000

4.6. Logikai feltételek használata

4.6.1. Hirdetés megtervezése

Feladat leírása: Egy cég marketing csapata hirdetési programot szervez. Különböző hirdetési formák közül kell választani úgy, hogy a hirdetések elindítása beleférjen a cég költségkeretébe, és az olvasók / nézők száma maximális legyen. A feladat egyszerűsítése miatt figyelmen kívül hagytam azt a tényezőt, hogy ugyanazon emberek több hirdetési forma által is találkozhatnak a cég reklámjával.
Az I tömb elemei poztiív egész számok 1-től 6-ig, mivel 6 db hirdetési forma közül lehet választani.
Ismert paraméterek:

- olvaso: Minden hirdetési módhoz adott, hány olvasót / nézőt ér el.
- koltseg: A hirdetési formákhoz tartozó költség
- tervezok: A hirdetés megtervezéséhez szükséges munkaerő fő \óra mértékegységben megadva
- eladok: A hirdetés eladásához szükséges munkaerő fő / óra mértékegységben megadva
- hird_forma: Az I tömb elemeihez hozzárendeli, hogy melyik hirdetési formát jelöli.

Változó:

- x: $x_i := \begin{cases} 1 & \text{ha az i. hirdetési formát használjuk} \\ 0 & \text{különben} \end{cases}$

Célfüggvény:

- olvaso: Maximalizálni akarjuk az olvasók\nézők számát.

Változót korlátozó feltételek:

- koltseg_max: Meg van adva a cég költségkerete, ezt nem léphetjük túl
- tervezok_max: Adott a hirdetés megtervezésére szolgáló munkaerő, ennél többet nem használhatunk fel.
- eladok_max: Adott a hirdetés eladására szolgáló munkaerő, ennél többet nem használhatunk fel.
- logika_1: A cég nem hirdethet egyszerre a kereskedelmi- és bulvár magazinban
- logika_2: Ha a promóciós kampány megvalósul, akkor szükséges a rádióban vagy a bulvár magazinban hirdetést feladni.
  Lehetséges kimenetelek:

	x_4	x_5	x_6	$x_4 + x_5 - x_6$
Megengedett értékek	1	1	1	1
	0	0	0	0
	1	0	0	0
	0	1	0	1
	1	1	0	2
	1	0	1	0
	0	1	1	0
Nem megengedett érték	0	0	1	-1

Megoldás:

```
set I;

param olvaso {i in I};
param koltseg {i in I};
param tervezok {i in I};
param eladok {i in I};
param hird_forma {i in I} symbolic;

var x{i in I} binary;

maximize olvasok: sum {i in I} x[i]*olvaso[i];

koltseg_max: sum {i in I} koltseg[i]*x[i]<=1800000;
tervezo_max: sum {i in I} tervezok[i]*x[i]<=1500;
```

```
eladok_max: sum {i in I} eladok[i]*x[i]<=1200;
logika_1: x[2]+x[5]<=1;
logika_2: x[4]+x[5]-x[6]>=0;

solve;

printf"\n" ;
printf"Kivalasztott hirdetesi formak:";
printf"\n";
for{i in I}
{printf(if(x[i]=1) then "%s \n" else ""), hird_forma[i];
}
printf"\n";
printf "Olvasok/nezok szama: ";
printf olvasok;
printf"\n";

data;
set I:=1 2 3 4 5 6;

param olvaso:= 1       1000000
               2       200000
               3       300000
               4       400000
               5       450000
               6       450000;

param koltseg:=1       500000
               2       150000
               3       300000
               4       250000
               5       250000
               6       100000;

param tervezok:=1      700
                2      250
                3      200
                4      200
                5      300
                6      400;
```

```
param eladok:= 1       200
               2       100
               3       100
               4       100
               5       100
               6       1000;

param hird_forma:=  1    TV
                    2    Kereskedelmi_magazin
                    3    Ujsag
                    4    Radio
                    5    Bulvar_magazin
                    6    Promocios_kampany;

end;
```

Kivalasztott hirdetesi formak:
TV
Ujsag
Radio
Bulvar_magazin

Olvasok/nezok szama: 2150000

4.6.2. Olajkutak fúrása

Feladat leírása: Egy olajfúró vállalatnak 10 helyből 5-öt kell kiválasztania úgy, hogy maximalizálja a bevételét.
Az I tömb elemei 1-től 10-ig pozitív egész számok, melyek a lehetséges olajkutak helyeinek felelnek meg.
Ismert paraméterek:

- koltseg: a fúrások költségeit adja meg millió Ft mértékegységben
- bevetel: Az olajkutak várható bevetele millió Ft \3 év mértékegységben

Változó:

- x: $x_i := \begin{cases} 1 & \text{ha az i. helyen fúrunk kutat} \\ 0 & \text{különben} \end{cases}$

Célfüggvény:

- bevetel_ossz: Az összbevételt növelik a kutakból származó bevételek, és csökkentik a kutak fúrási költségei

Változót korlátozó feltételek

- max_5: Maximum 5 helyen fúrhatunk
- logika_1: Ha a 3. helyen fúrunk, akkor az 5. helyen nem fúrhatunk
- logika_2: Ha a 4. helyen fúrunk, akkor az 5. helyen nem fúrhatunk
- logika_3: Az 5., 6., 7, és 8, helyek közül legfeljebb csak 2 helyen fúrhatunk
- logika_4: ha az 1. és 7. helyen fúrunk, akkor a 8, helyen már nem fúrhatunk kutat

Megoldás:

```
set I;

param koltseg {i in I};
param bevetel{i in I};

var x {i in I} binary;

maximize bevetelossz: sum {i in I} ↩
   (bevetel[i]*x[i]-koltseg[i]*x[i]);

max_5: sum {i in I} x[i]<=5;
logika_1 :  x[3]+x[5]<=1;
logika_2: x[4]+x[5]<=1;
logika_3 : x[5]+x[6]+x[7]+x[8]<=2;
logika_4 : x[1]+x[7]+x[8]<=2;
solve;

printf"\n";
printf "Valsztott helyek:";
printf "\n";
for{i in I}
{printf(if(x[i]>0) then "%s \n " else ""), i;
}
printf"\n";
printf "Bevetel: ";
printf bevetelossz;
```

```
printf"\n";

data;

set I:=1 2 3 4 5 6 7 8 9 10;

param koltseg:= 1    520
                2    540
                3    480
                4    515
                5    537
                6    525
                7    497
                8    502
                9    552
                10   519;

param bevetel:=  1    654
                 2    627
                 3    612
                 4    644
                 5    632
                 6    617
                 7    593
                 8    652
                 9    642
                 10   613;

end;
```

Valasztott helyek:
1
3
4
8
10

Bevetel: 639

4.7. Befektetési lehetőségek

Feladat leírása: Egy vállalkozás vezetője, aki 10 féle erőforrással rendelkezik a nyereségét szeretné befektetni. 6 befektetési lehetőség közül választhat, úgy hogy a felhasználható erőforrások mennyisége korlátozott, és maximalizálni szeretné a bevételét.
Az I tömb elemei a befektetési lehetőségeket jelöli, a J tömbbé pedig az elérhető és szükséges erőforrásokat.
Ismert paraméterek:

- bevetel: A befektetések várható nyeresége millió Ft mértékegységben megadva
- szukseges: Megadja, hogy melyik befektetéshez mennyi erőforrás felhasználására van szükség
- elerheto: Az erőforrásokból elérhető mennyiséget jelöli.
- megnevezes: A különböző befektetések rövid megnevezései

Változó:

- x: $x_i := \begin{cases} 1 & \text{ha az i. befektetés megvalósul} \\ 0 & \text{különben} \end{cases}$

Célfüggvény:

- osszebvetel: maximalizáljuk a befektetésekből származó bevételt

Korlátozó feltételek:

- max_elerheto: Összesen csak annyi erőforrást használhatunk fel, amennyi rendelkezésre áll.

Megoldás:

```
set I;
set J;

param bevetel {i in I};
param szukseges {i in I, j in J};
param elerheto {j in J};
param megnevezes {i in I} symbolic;

var x {i in I} binary;

maximize osszbevetel: sum {i in I} x[i]*bevetel[i];
```

```
max_elerheto {i in I}: sum {j in J} ↩
   szukseges[i,j]*x[i]<=elerheto[i];

solve;

printf"\n";
printf "Kivalasztott befektetesek: \n";
for {i in I}
{printf(if (x[i]>0) then " %s \n" else ""), megnevezes[i];
}
printf "\n Varhato bevetel:  ";
printf osszbevetel;
printf"\n";

data;

set J:= 1 2 3 4 5 6 7 8 9 10;
set I:=1 2 3 4 5 6;

param bevetel:= 1   100
                2   79
                3   68
                4   132
                5   94
                6   85;

param szukseges:
      1     2    3    4    5    6    7    8     9     10:=
 1    10    4    0    2    0    0    9    8     2     6
 2    5     9    1    9    0    8    4    12    9     5
 3    7     3    3    11   2    3    1    7     2     3
 4    3     6    2    0    2    2    9    2     3     14
 5    0     2    4    7    1    5    10   3     7     2
 6    2     4    2    4    7    0    1    0     4     4    ;
```

```
param elerheto:=1      60
                2      40
                3      31
                4      76
                5      54
                6      34
                7      42
                8      36
                9      41
                10     67;

param megnevezes:= 1     ertekpapirok
                   2     gyogyszer_ceg
                   3     informatikai_ceg
                   4     ingatlanok_vasarlasa
                   5     gyogyfurdok
                   6     szalloda;
end;
```

Kivalasztott befektetesek
ertekpapirok
ingatlanok_vasarlasa
gyogyfurdok
szalloda
Varhato bevetel: 411

Feladat más értelmezései:

- Befektetési periódusok: Az erőforrásokra tekinthetünk befektetési periódusként is. Ebben az esetben a különböző periódusokban különböző a befektetés érték, mivel felhasználhatók a korábbi befektetések eredménye is.

- Hátizsák feladat: Ezzel a modellel a hátizsák feladat is felírható, amelyben adott egy hátizsák véges teherbírással, és célunk különböző értéktárgyak berakása a hátizsákba úgy, hogy a teherbírást ne lépjük túl, és a benne lévő tárgyak értéke maximális legyen. Ebben az esetben az erőforrásokat kihagyjuk a feladatból. A befektetések az értéktárgyaknak felelnek meg, a bevetel paraméter a tárgyak értékeit jelölik, szukseges parameter helyett az értéktárgyak súlyát adjuk meg és az elerheto parameter pedig a táska teherbírását jelöli.

5. Hivatkozások

[1] S. Bradley, A. Hax, T. Magnanti, *Applied Mathematical Programming*, Addison-Wesley (1977) ISBN-13: 9780201004649, Chapter 09

[2] Leo Liberti,*Problems and exercises in Operations Research* (2010), Chapter 04

[3] Fábián Csaba, Király Tamás, Papp Olga, *Operációkutatás jegyzet* (2015)

[4] Jayaswal, Sachin, *A comparative study of tabu search and simulated annealing for traveling salesman problem. Department of Management Sciences, Univerisity of Waterloo, Project Report* (2012)

[5] Szőnyi Tamás, *Hamilton-utak , Hamilton-körök*

[6] Freud Róbert,*Lineáris algebra* (2014)

[7] Clifford Stein (Professor at Columbia University),*Airline Scheduling Case Study*

[8] http://www.tavolsag.hu