Linearni problem prevoza

Dragana Božović in Andrej Taranenko

→ **Tovarna čokolad skladišči svoje izdelke na različnih lokacijah v Sloveniji, te izdelke pa želijo dostaviti v več trgovin, prav tako na različnih lokacijah. V času krize želijo pri razvozu svojih izdelkov čim več prihraniti. Kako naj tovarna dostavi svoje izdelke trgovinam tako, da bo vsaka trgovina prejela vsaj naročeno število izdelkov, da bo skupno število razvoženih izdelkov največ enako zalogi leteh in da bo tovarno razvoz izdelkov iz skladišč v trgovine stal čim manj? Na ta vprašanja bomo odgovorili v tem članku.**

Linearno programiranje in problem prevoza

Problemi linearnega programiranja so optimizacijski problemi, v katerih želimo optimizirati določeno *ciljno funkcijo*, pri čemer moramo zadoščati podanim pogojem oziroma omejitvam. V tovrstnih problemih so pogoji in ciljna funkcija vedno linearne funkcije optimizacijskih spremenljivk.

Pobližje bomo spoznali *problem prevoza*, znan tudi pod imenom *transportni problem*. Problem prevoza je optimizacijski problem, v katerem največkrat želimo minimizirati stroške prevoza. Za lažjo predstavo vzemimo našo tovarno čokolad. Recimo, da ima ta tovarna na različnih lokacijah po Sloveniji m skladišč in n trgovin. Posamezni izdelek lahko iz nekega skladišča dostavimo do vsake trgovine. V vsakem skladišču imamo natančno določeno količino oziroma zalogo izdelkov, ki so v danem času skladiščeni (torej, na voljo). Vsaka trgovina v danem času naroči točno določeno količino izdelkov (povpraševanje trgovine). Še več, za vsako skladišče je znano, koliko stane prevoz enote izdelka do vsake izmed trgovin. Cena je običajno odvisna od razdalje med skladiščem in trgovino ter od prevoznega sredstva, ki ga uporabimo. Predpostavimo, da so cene prevoza posamezne enote izdelka (npr. karton čokolad) od skladišča do trgovine znane in so celoštevilske.

Označimo omenjene količine na naslednji način:

- Število s_i predstavlja zalogo (enot) izdelka v skladišču i, kjer je $i = 1, 2, \ldots, m$.
- Število d_j predstavlja, koliko enot izdelka naroča trgovina j, kjer je $j = 1, 2, \ldots, n$.
- S številom $c_{i,j}$ označimo, koliko stane prevoz ene enote izdelka iz skladišča i v trgovino j, za $i = 1, 2, \ldots, m$ in $j = 1, 2, \ldots, n$.
- S številom $x_{i,j}$ označimo, koliko enot izdelka peljemo iz skladišča i do trgovine j, za $i = 1, 2, \ldots, m$ in $j = 1, 2, \ldots, n$.

Cilj je določiti, kako razvoziti izdelke med skladišči in trgovinami tako, da bo razvoz ustrezal zalogam (ponudbam) in povpraševanjem, pri čemer bodo skupni stroški najmanjši možni.

Opišimo zastavljeni problem še matematično. S *ciljno funkcijo* zapišimo skupne stroške, ki jih želimo minimizirati. Ceno prevoza iz skladišča i do trgovine j lahko torej izračunamo kot $c_{i,j} \cdot x_{i,j}$. Skupno ceno razvoza iz vseh skladišč do vseh trgovin dobimo tako, da seštejemo stroške za vse možne pare vrednosti števil i in j. Naša ciljna funkcija z je torej (linearna) funkcija:

- $$z = \sum_{i=1}^{m} \sum_{j=1}^{n} (c_{i,j} \cdot x_{i,j}).$$

Seveda želimo določiti vrednosti $x_{i,j}$ tako, da bo funkcija z imela za te vrednosti čim manjšo vrednost (želimo minimizirati z).

Opišimo še *omejitve*, ki se v problemu prevoza naravno pojavijo. Iz skladišča i lahko pošljemo izdelke v različne trgovine. Število vseh enot izdelkov, ki jih iz skladišča i pošljemo, je torej enako vsoti vseh količin, ki smo jih iz tega skladišča razvozili do vseh trgovin, torej od količine enot iz skladišča i, ki jo peljemo do trgovine 1 ($x_{i,1}$), do trgovine 2 ($x_{i,2}$), in tako naprej, vse do trgovine n ($x_{i,n}$). Zapisano drugače, iz skladišča i razvozimo skupno $x_{i,1} + x_{i,2} + \ldots + x_{i,n}$ enot izdelka. Zapisano s simbolom za vsoto, je to

$\sum_{j=1}^{n} x_{i,j}$. Ker ima (vsako) skladišče i omejeno zalogo s_i, iz tega skladišča ne moremo poslati več izdelkov, kot jih je na zalogi. To omejitev lahko za vsako skladišče $i = 1, \ldots, m$ izrazimo na naslednji način:

- $$\sum_{j=1}^{n} x_{i,j} \leq s_i.$$

Poglejmo še trgovino j. V to trgovino prispejo izdelki iz vseh skladišč (količina je lahko tudi 0, kar pomeni, da iz skladišča v to trgovino nismo poslali izdelka). Torej je skupna količina enot izdelka, ki jo trgovina prejme, enaka $x_{1,j}+x_{2,j}+\ldots+x_{m,j}$. Znova, zapisano krajše, je ta količina $\sum_{i=1}^{m} x_{i,j}$. Ker je skupno povpraševanje trgovine j enako d_j, potem celotna količina dostavljenih izdelkov ne sme biti manjša od d_j (lahko pa trgovina skupno prejme več, kot je naročila). Ta pogoj lahko za vsako trgovino $j = 1, \ldots, n$ zapišemo kot

- $$\sum_{i=1}^{m} x_{i,j} \geq d_j.$$

Ker lahko iz skladišča do trgovine pošljemo 0 ali več enot izdelkov, morajo biti vsi $x_{i,j}$ nenegativna cela števila (enot ne moremo še bolj razdrobiti).

Problem prevoza lahko sedaj formalno zapišemo na naslednji način [2]:

- minimiziraj $$z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{i,j} x_{i,j} \qquad (1)$$

pri pogojih:

$$\sum_{j=1}^{n} x_{i,j} \leq s_i \qquad \text{za } i = 1, \ldots, m$$

$$\sum_{i=1}^{m} x_{i,j} \geq d_j \qquad \text{za } j = 1, \ldots, n$$

$$x_{i,j} \geq 0 \qquad \text{za } i = 1, \ldots, m \text{ in } j = 1, \ldots, n$$

Kadar je skupna količina zaloge vseh skladišč enaka skupni količini povpraševanj vseh trgovin, pravimo, da je problem prevoza *uravnotežen*. Zapisano drugače, problem prevoza je uravnotežen, če velja

- $$\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j.$$

Poglejmo si zadevo še na konkretnem primeru, ki je povzet po [1]. Naša tovarna čokolad ima v Sloveniji tri različna skladišča. Mesečno s svojimi izdelki oskrbuje štiri trgovine, ki se nahajajo v različnih krajih. Poznamo zalogo skladišč, povpraševanje trgovin in transportne stroške na enoto izdelka iz vsakega skladišča do vsake trgovine. Kako naj tovarna razvozi svoje izdelke iz skladišč do trgovin tako, da bodo transportni stroški najmanjši možni?

Podatki tega problema so prikazani v tabeli 1. Števila, zapisana s krepko, predstavljajo transportne stroške. Tako vidimo, da je cena prevoza enote izdelka iz skladišča 1 do trgovine 1 enaka 10, prevoz iz skladišča 2 do trgovine 3 stane na enoto 9, strošek prevoza ene enote izdelka iz skladišča 3 do trgovine 1 pa je brezplačen (enak 0). Opazimo lahko tudi, da je ta problem prevoza uravnotežen. Problem bomo rešili s pomočjo programa (reševalnika) za reševanje problemov linearnega programiranja, za to pa moramo spoznati jezik AMPL, s katerim opišemo naš primer.

	trgovine				
skladišče	1	2	3	4	zaloga skladišča
1	**10**	**0**	**20**	**11**	20
2	**12**	**7**	**9**	**20**	25
3	**0**	**14**	**16**	**18**	15
povpraševanje	10	15	15	20	

TABELA 1.
Podatki za razvoz.

Jezik AMPL

Problem bomo rešili s pomočjo jezika AMPL (angl. *A Mathematical Programming Language*). AMPL je jezik za modeliranje v matematični optimizaciji. Uporabljamo ga za opisovanje in reševanje problemov z veliko zahtevnostjo ter obsežnim matematičnim računanjem. Razvili so ga Robert Fourer, David Gay and Brian Kernighan v Bell Laboratories. AMPL podpira desetine različnih reševalnikov problemov, ki so tako odprto-kodni kot komercialni.

Posebna prednost jezika AMPL je podobnost njegove sintakse z matematičnim zapisom optimizacijskih problemov. Ta prednost omogoča zelo čitljivo in jedrnato opredelitev problemov na področju opti-

mizacije. Mnogi sodobni reševalniki, ki so dostopni na strežniku NEOS[1], sprejemajo za vhod probleme, zapisane v jeziku AMPL. Glede na statistike strežnika NEOS je AMPL najbolj priljubljen format za predstavitev problemov matematičnega programiranja [3].

Preden problem rešimo, ga moramo zapisati v ustrezni obliki, ki predstavlja vhod (vhodne podatke) za reševalnik. Vhod je v AMPL-u sestavljen iz treh datotek, in sicer `.mod`, `.dat` in `.txt`.

Ko pričnemo s programiranjem, najprej ustvarimo datoteko `.mod`, v kateri podamo ciljno funkcijo in omejitve našega problema. Na začetku datoteke najprej definiramo množice, ki so potrebne za reševanje problema. Množico, vključno z njenimi elementi, zapišemo na naslednji način, pri čemer v zavitih oklepajih podamo elemente množice:

- `set A := {`$a_1, a_2, \ldots, a_k$`};`

Definiramo lahko parametre, ki predstavljajo podatke, podane na začetku reševanja problema. Parameter definiramo kot

- `param Ime {A};`

Pri tem `A` predstavlja množico, na katero se sam parameter navezuje. Parameter se lahko navezuje tudi na več množic (večdimenzionalni parameter), ki jih podamo v zavitem oklepaju ter jih ločimo z vejico.

Naslednji korak je definicija spremenljivk našega problema. Spremenljivko definiramo kot

- `var Ime {a in A};`

Tudi spremenljivka se lahko navezuje na več množic, ki jih zapišemo v zavitih oklepajih ter ločimo z vejico. Ciljno funkcijo in želeno optimizacijo podamo kot

- `minimize/maximize Ime: funkcija;`

Omejitve problema pa zapišemo kot

- `subject to Ime: omejitev;`

Če se omejitev navezuje na določeno množico, to zapišemo kot

- `subject to Ime {a in A}: omejitev;`

[1]Seznam najdemo na naslovu `http://www.neos-server.org/neos/solvers/index.html`, preverjeno 3. 10. 2013

Problem iz našega primera bi zapisali na način, kot je prikazan v izpisu datoteke 1. Najprej definiramo množici $M = \{1, 2, 3\}$ in $N = \{1, 2, 3, 4\}$. Nato napovemo, da bomo imeli tri parametre: parameter c bo matrika dimenzije 3×4 (indeksi so vzeti iz množic M in N) in predstavlja matriko cen, parameter s predstavlja vektor zalog skladišč, d pa vektor povpraševanj trgovin. Vrstica `var x {i in M,j in N} >= 0;` deklarira optimizacijske spremenljivke $x_{i,j}$ in pove, da morajo biti nenegativne. Zadnje tri vrstice podajo ciljno funkcijo, ki jo minimiziramo, ter omejitve optimizacije.

Izpis datoteke 1 AMPL model (.mod)

```
set M := {1..3};
set N := {1..4};
param c {M,N};
param s {i in M};
param d {j in N};
var x {i in M,j in N} >= 0;
minimize sudoku:
    sum {i in M, j in N} c[i,j]*x[i,j];
subject to izvori {i in M}:
    sum {j in N} x[i,j] <= s[i];
subject to povpraševanje {j in N}:
    sum {i in M} x[i,j] >= d[j];
```

V podatkovni datoteki `.dat` moramo določiti vrednosti parametrov, ki smo jih napovedali v modelu. Parametre, ki predstavljajo vektorje, zapišemo na naslednji način:

- `param imeParametra :=`
  `indeks1 vrednost1 indeks2 vrednost2 ...;`

Če parameter predstavlja matriko, pa pred `:=` in za parametrom zapišemo indekse stolpcev, za tem simbolom pa zapišemo indeks vrstice in naštejemo vse elemente, ločene s presledkom. Na koncu zadnje vrstice ne smemo pozabiti na podpičje. Pozorni moramo biti tudi na to, da indekse iz množice, ki smo jo v deklaraciji parametra podali kot prvo, damo v vrstice, elemente druge množice pa v stolpce.

Datoteka s podatki za naš primer je prikazana v izpisu datoteke 2. Najprej definiramo parametra s in d. Kot že vemo, s predstavlja vektor zalog skladišč, d pa vektor povpraševanj trgovin. Zapišemo ju tako, da najprej podamo indeks v vektorju, takoj za njim pa vrednost, ki je na tem mestu v vektorju. To storimo za vse koordinate vektorja. Vse to zapišemo v

eni vrstici, pri čemer številke ločimo s presledki. V prvi vrstici za vektor s vidimo, da je vrednost na prvi koordinati je 20, na drugi 25 in na tretji 15. Podobno velja za vektor d. Nato definiramo še parameter c oziroma matriko cen, ki je dimenzij 3×4. Najprej je treba zapisati indekse vseh stolpcev, v našem primeru naravna števila od ena do štiri (kot je zapisano v četrti vrstici). Nato zapišemo vrstice. Najprej zapišemo številko vrstice, nato naštejemo vrednosti, ki se v tej vrstici nahajajo. To storimo za vse vrstice, ki jih imamo, pri čemer številke ločimo s presledki. Torej najprej za prvo vrstico (1 10 0 20 11), nato drugo (2 12 7 9 20) in na koncu še tretjo vrstico (3 0 14 16 18).

Izpis datoteke 2 AMPL model (.dat)

```
param s := 1 20 2 25 3 15;
param d := 1 10 2 15 3 15 4 20;
param c:
1 2 3 4 :=
1 10 0 20 11
2 12 7 9 20
3 0 14 16 18;
```

Reševalniku moramo podati tudi ukaze, kaj naj z modelom in podatki stori. S stavkom `solve;` mu povemo, naj problem reši, s stavkom `display x;` pa, da naj izpiše rezultat, torej končne vrednosti optimizacijskih spremenljivk `x` (matriko). Ukazna datoteka je prikazana v izpisu datoteke 3.

Izpis datoteke 3 AMPL model (.txt)

```
solve;
display x;
```

Nato naložimo vse tri datoteke in počakamo na rešitev optimizacijske naloge [4]. NEOS reševalnik (Mixed Integer Linear Programming - scip[AMPL Input]) izpiše naslednji rezultat:

```
x  :=
1  1   0
1  2   5
1  3   0
1  4  15
2  1   0
2  2  10
2  3  15
2  4   0
3  1  10
3  2   0
3  3   0
3  4   5
;
```

To rešitev lahko predstavimo z naslednjo tabelo, pri čemer prva vrstica predstavlja oznake trgovin, prvi stolpec pa oznake skladišč.

	1	2	3	4
1	0	5	0	15
2	0	10	15	0
3	10	0	0	5

Torej so stroški prevoza našega problema najmanjši v primeru, ko imamo naslednje vrednosti optimizacijskih spremenljivk:

- $x_{1,2} = 5,\ x_{1,4} = 15,\ x_{2,2} = 10,$
  $x_{2,3} = 15,\ x_{3,1} = 10,\ x_{3,4} = 15,$

vse ostale vrednosti $x_{i,j}$, ki jih tukaj nismo našteli, imajo vrednost 0. Najmanjši stroški prevoza za naš primer znašajo

- $5 \cdot 0 + 15 \cdot 11 + 10 \cdot 7 + 15 \cdot 9 + 10 \cdot 0 + 5 \cdot 18 = 460.$

Literatura

[1] J. F. Rayman, Transportation Problems (online), 3. 10. 2013. http://personal.maths.surrey.ac.uk/st/J.F/chapter7.pdf

[2] The Transportation Problem: LP Formulations (online), 3. 10. 2013. http://www.utdallas.edu/~scniu/OPRE-6201/documents/TP1-Formulation.pdf

[3] Wikipedia: AMPL (online), 3. 10. 2013. http://en.wikipedia.org/wiki/AMPL

[4] AMPL (online), 3. 10. 2013. http://um.fnm.uni-mb.si/tiki-download_wiki_attachment.php?attId=241&page=02.11.2011\%3A\%20Linearni\%20program.\%20Dual.\%20Simpleksna\%20metoda

×××