VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Optimizavimo metodai

Laboratorinis darbas nr. 4

Tiesinis programavimas

Tomas Giedraitis
VU MIF Informatika
3 kursas 3 grupė

Vilnius
2019

Tiesinio programavimo uždavinys

Variantas 1

$$min(2x_1 - 3x_2 - 5x_4)$$

$$-x_1 + x_2 - x_3 - x_4 \le 8$$

$$2x_1 + 4x_2 \le 10$$

$$x_3 + x_4 \le 3$$

$$x_i \ge 0$$

Duotas uždavinys matriciniu pavidalu:

$$min\,CX \quad ,$$
$$AX \le B \quad ,$$
$$X \ge 0 \quad ,$$

kur $C = \begin{pmatrix} 2 & -3 & 0 & -5 \end{pmatrix}$,

$$A = \begin{pmatrix} -1 & 1 & -1 & -1 \\ 2 & 4 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

$$B = \begin{pmatrix} 8 & 10 & 3 \end{pmatrix}$$

Suvedame uždavinį į standartinę formą:

$$min(2x_1 - 3x_2 - 5x_4)$$

$$-x_1 + x_2 - x_3 - x_4 + s1 = 8$$

$$2x_1 + 4x_2 + s2 = 10$$

$$x_3 + x_4 + s3 = 3$$

$x_i \ge 0, s_i \ge 0$, čia s_i - fiktyvūs kintamieji.

Taip pat perrašome matriciniu pavidalu:

$$min\,CX \quad ,$$
$$AX = B \quad ,$$
$$X \ge 0 \quad ,$$

kur $C = \begin{pmatrix} 2 & -3 & 0 & -5 & 0 & 0 & 0 \end{pmatrix}$,

$$A = \begin{pmatrix} -1 & 1 & -1 & -1 & 1 & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix},$$

$$B = \begin{pmatrix} 8 & 10 & 3 \end{pmatrix}$$

Uždavinys sprendžiamas naudojant suprogramuotą simplekso algoritmą tiesiniam programavimui.

Algoritmo veikimo principas:

Susivedame matricas C, A ir B į vieną matricą M (žr. tarpines programos išvestis). Aprašydami algoritmą, C, A ir B laikysime tuos segmentus (matricas) matricoje M, kurie prieš tai priklausė vienai iš šių matricų (C,A,B), prieš sujungus jas į matricą M:

$$M = \begin{pmatrix} 2 & -3 & 0 & -5 & 0 & 0 & 0 & 0 \\ -1 & 1 & -1 & -1 & 1 & 0 & 0 & 8 \\ 2 & 4 & 0 & 0 & 0 & 1 & 0 & 10 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 3 \end{pmatrix}$$

Baziniai stulpeliai pradžioje turi indeksus *beta = [5,6,7]*, nes nuo šių stulpelių prasideda fiktyvūs kintamieji (*beta* vadinama baze). Jie atitinka trijų dimensijų vienetinę matricą $E = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$, kur atitinkamai 5-as stulpelis atitinka pirmąjį vienetinės matricos stulpelį $k=1$, 6-as atitinka $k=2$, 7-as – $k=3$.

Vykdome ciklą:

> Surandame mažiausią neigiamą reikšmę matricoje C *(=-5)*, ir fiksuojame stulpelio numerį i *(i=4)*. Šis stulpelis bus naujos bazės stulpelis. Tuomet skaičiuojame *lambda* vektorių, gaunamą kiekvieną B matricos (apribojimų matricos) elementą padalinus iš skaičiaus matricoje A toje pačioje eilutėje, ir esančio stulpelyje i (dalybą iš nulio galime traktuoti kaip 0, nes svarbios tik teigiamos *lambda* vektoriaus reikšmės):
>
> $$lambda = \begin{pmatrix} \frac{8}{-1} & \frac{10}{0} & \frac{3}{1} \end{pmatrix}$$
>
> Surandame mažiausią tegiamą reikšmę *lambda* vektoriuje *(=3)*. Jos numeris *lambda* vektoriuje *(=3)* nurodo, kelintas iš trijų (šiuo atveju) bazės stulpelių patampa stulpelis i (t.y. gaunamas skaičius k *(k=3)*, kurio reikšmė šiuo atveju yra nuo *1* iki *3*). Atitinkamai, pagal indeksą k, stulpelis i bandomas suvesti į vienetinės matricos atitinkamą indekso stulpelį (neįtraukiant pirmosios M matricos eilutės – buvusios C matricos), t.y. atitinkamoje eilutėje (pagal indeksą k) turi būti vienetas, o kitose eilutėse – nuliai). Šiuo atveju: $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$, kadangi $k = 3$. Taip pat, mažiausia neigiama reikšmė pirmoje matricos eilutėje, i-ajame stulpelyje *(=-5)* turi būti pakeista į *0*. Visa tai atliekama naudojant elementariuosius pertvarkymus (Gauso metodą).

Šis ciklas kartojamas, kol nebelieka neigiamų reikšmių matricoje *C*. Uždavinio atsakymus gauname paskutiniame matricos stulpelyje. Funkcijos minimumą nurodo pirmas šio stulpelio elementas, tik papildomai dar padauginama iš *-1*, nes kitaip gautumėme maksimumą. Argumentų (kintamųjų) reikšmės (įskaitant ir fiktyviųjų) seka nuo antro elemento iki stulpelio pabaigos, t.y. ten, kur pradžioje sudarius matricą M buvo matricos B elementai.

Belieka nustatyti, kurios reikšmės priklauso kuriam argumentui. Tam pasitelkiame paskutinėje ciklo iteracijoje suformuotą bazę (*beta*), kurioje dabar yra argumentų reikšmių indeksai:

Tarkime, paskutinis matricos stulpelis paskutinėje iteracijoje:

$$\begin{pmatrix} 22.5 \\ 8.5 \\ 2.5 \\ 3 \end{pmatrix} .$$

F-jos minimumas = $22.5*(-1)=-22.5$

Argumentų reikšmės = [8.5, 2.5, 3]

Gauta bazė *beta* = [5,2,4]

Taigi,

8.5 bus 5-ojo argumento reikšmė (pagal bazę *beta*)
2.5 – 2-ojo,
3 – 4-ojo.

Visi kiti argumentai įgaus nulines reikšmes.

Gaunamas argumentų vektorius = [0, 2.5, 0, 3, 8.5, 0, 0], kurį išskaidome į X (kintamųjų vektorių) ir S (fiktyvių kintamųjų vektorių):

X = [0, 2.5, 0, 3],

S = [8.5, 0, 0].

Tarpinės programos išvestys:

M=

2	-3	0	-5	0	0	0	0
-1	1	-1	-1	1	0	0	8
2	4	0	0	0	1	0	10
0	0	1	1	0	0	1	3

beta=

5 6 7

M=

2	-3	5	0	0	0	5	15
1	-1	0	0	-1	-0	-1	-11
2	4	0	0	0	1	0	10
0	0	1	1	0	0	1	3

beta=

5 6 4

M=

3.5	0	5	0	0	0.75	5	22.5
-1.5	0	-0	-0	1	-0.25	1	8.5
0.5	1	0	0	0	0.25	0	2.5
0	0	1	1	0	0	1	3

beta=

5 2 4

X=

0 2.5 0 3

S=

8.5 0 0

Fmin=

-22.500

Uždavinio atsakymas:

rastas f-jos minimumas: *Fmin*=-22.5
argumentų rinkinys: X = [0, 2.5, 0, 3]
bazė: *beta* = [5, 2, 4]

Variantas 2

Pakeisime apribojimų dešinės pusės kontantas į 1, 4, 7.

Tuomet matrica $B = \begin{pmatrix} 1 & 4 & 7 \end{pmatrix}$

Gauname naują uždavinį, kurį vėlgi sprendžiame naudojant tą patį algoritmą.

Tarpinės programos išvestys:

```
M=
  2  -3  0  -5  0  0  0  0
  -1  1  -1  -1  1  0  0  1
  2  4  0  0  0  1  0  4
  0  0  1  1  0  0  1  7
beta=
  5  6  7

M=
  2  -3  5  0  0  0  5  35
  1  -1  0  0  -1  -0  -1  -8
  2  4  0  0  0  1  0  4
  0  0  1  1  0  0  1  7
beta=
  5  6  4

M=
  3.5  0  5  0  0  0.75  5  38
  -1.5  0  -0  -0  1  -0.25  1  7
  0.5  1  0  0  0  0.25  0  1
  0  0  1  1  0  0  1  7
beta=
  5  2  4

X=
  0  1  0  7
S=
  7  0  0
Fmin=
-38
```

Uždavinio atsakymas:

rastas f-jos minimumas: *Fmin*=-38
argumentų rinkinys: X = [0, 1, 0, 7]
bazė: *beta* = [5, 2, 4]

Simplekso algoritmas tiesiniams uždaviniams

```
function LinearProgramming
% veikia tik kai apribojimai yra "<="

% =========== vartotojo ivedami duomenys =============
% kintamuju skaicius
n = 4;
% koeficientai salia kintamuju
kint = [2, -3, 0, -5];
% apribojimu skaicius
a_n = 3;
% apribojimu kairiosios puses
ap_LHS = [-1  1 -1 -1
          2  4  0  0
          0  0  1  1];
% apribojimu desiniosios puses
ap_RHS = [ 1
          4
          7 ];
% ==================================================

M = zeros(a_n+1,n+a_n+1);
M(1,1:n) = kint;
M(2:end,1:n) = ap_LHS;
M(2:end,n+1:n+a_n) = eye(a_n);
M(2:end,end) = ap_RHS;

% last row
lastRow = size(M,1);
% last column
lastCol = size(M,2);

beta = [n+1, n+2, n+3];

while true

format short g;

disp('M=');
disp(M);
disp('beta=');
disp(beta);

% cl = lowest col number
[lowest,cl] = min(M(1:1,1:n));

if (lowest >= 0)
  break
endif

lambda = [];
```

```
for (row = 2:lastRow)
   lambda(end+1) = M(row,lastCol) ./ M(row, cl);
end

lambda(lambda < 0) = NaN;
[m,k_ind] = (min(lambda));
beta(k_ind) = cl;
base_row = k_ind + 1;

el = M(base_row,cl);
M(base_row, :) = M(base_row, :) ./ el;

for (row = 2:lastRow)
  el = M(row, cl);
  if (el ~= 0)
    M(row, :) = M(row, :) ./ el;
    if (row ~= base_row)
      M(row, :) = M(row, :) - M(base_row, :);
    endif
  endif
endfor

M(1, :) = M(1, :) - M(base_row, :) .* lowest;
endwhile

RES = [0,0,0,0,0,0,0];
B=M(2:lastRow,lastCol)
RES(beta) = B

X = RES(1:n);
S = RES(n+1:end);
Fmin = -M(1,lastCol);

disp("X=");
disp(X);
disp("S=");
disp(S);
disp("Fmin=");
disp(Fmin);

endfunction
```