Optimizavimo metodai

Užduoties „Tiesinis programavimas“ ataskaita

Autorius: Vilius Minkevičius, Programų Sistemos 4 kursas 2 grupė

Darbo eiga

1. Optimizavimo uždavinio formulavimas,
2. Simplekso algoritmo įgyvendinimas,
3. Algoritmo pritaikymas duotam uždaviniui ir pagal studento numerį suformuluotam,
4. Rezultatų palyginimas.

Užduoties formulavimas

1. Tikslo funkcija: **$f(X) = 2x_1 - 3x_2 - 5x_4$**
2. Uždavinys pagal sąlygą:
   min f(X),
   $-x_1 + x_2 - x_3 - x_4 \leq 8$,
   $2x_1 + 4x_2 \leq 10$,
   $x_3 + x_4 \leq 3$,
   $x_i \geq 0$.
3. Uždavinys, užrašytas matriciniu pavidalu standartinėje formoje:
   min f(X),
   $$\begin{pmatrix} -1 & 1 & -1 & -1 & 1 & 0 & 0 \\ 2 & 4 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} X = \begin{pmatrix} 8 \\ 10 \\ 3 \end{pmatrix},$$
   $x_i, c_i \geq 0$,
   Kintamieji:
   $X = (x_1, x_2, x_3, x_4, c_1, c_2, c_3)^T$, kur c_i yra fiktyvūs kintamieji.

Simplekso algoritmo įgyvendinimas

1. Sukūriau funkciją **„abaks“**, kuri atlieka atraminės bazės keitimą (toliau ABAKS). Ji pasiima matricą, kuri turi bazinius stulpelius, ir prideda nurodytą stulpelį į bazę. Yra nustatoma, kurį stulpelį reikia pašalinti iš bazės ir jis yra pašalinamas. Grąžinama matrica su nauja baze.
2. Atlikus ABAKS‘ą prireikė funkcijos, kuri nustatytų naujos bazės stulpelių numerius. Tam sukūriau **„get_base“** funkciją.
3. Minėtas funkcijas patalpinau į „while“ ciklą, kad algoritmas būtų vykdomas tol, kol yra bent vienas stulpelis, kurį verta įtraukti į bazę.

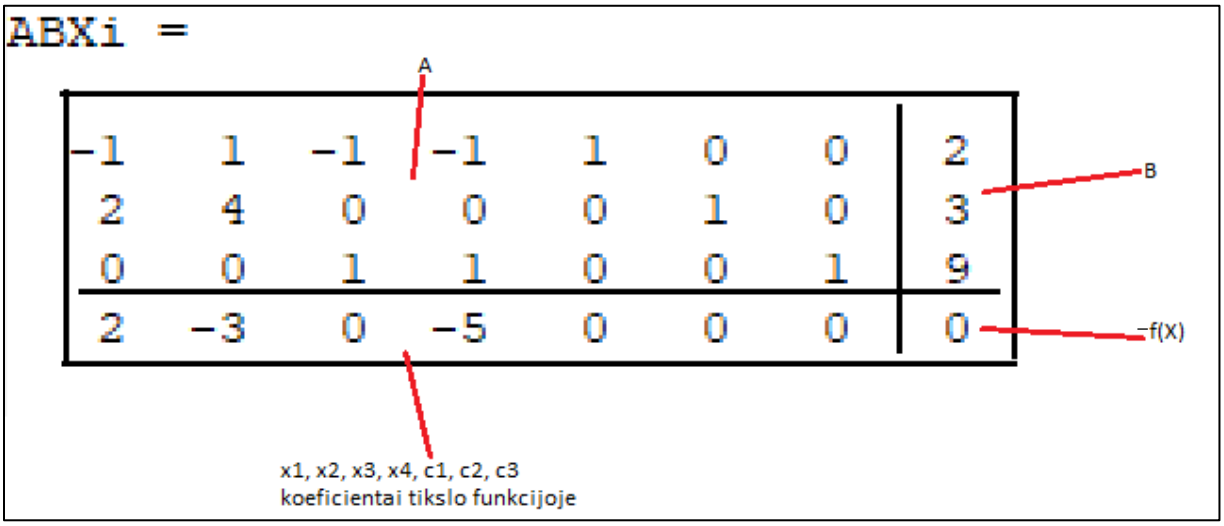
Algoritmo pritaikymas uždaviniams spręsti

1. Programos pradžioje apibrėžiau pradinius duomenis: tikslo funkciją, uždavinį matriciniu pavidalu, pradinės bazės stulpelius ir eilutę su tikslo funkcijos koeficientais ties X vektoriaus komponentais.

2. ABAKS‘u keičiamos matricos struktūra:

a. **N+1 stulpelių**, kur **N** yra kintamųjų vektoriaus ilgis. Pirmi N stulpelių yra tiesinės lygčių sistemos koeficientai (A, kairioji lygybių pusė), o paskutinis stulpelis – dešinioji lygybių pusė (B).
b. **M+1 eilučių**, kur **M** yra tiesinių lygčių kiekis sistemoje. Pirmos M eilučių yra iš uždavinio matricinio pavidalo. Paskutinėje eilutėje laikomi tikslo funkcijos koeficientai, kurie parodo, tikslo funkcijos pokyčio kryptį, įtraukus stulpelį į bazę (teigiama – padidės reikšmė, neigiama – sumažės). Paskutinės eilutės paskutinis elementas – tikslo funkcijos reikšmė (su minuso ženklu), skaičiavimo pradžioje lygi nuliui.

3. Norint modifikuoti sprendžiamą uždavinį, reikia pakeisti tarp pradinių duomenų esančius kintamuosius. Pavyzdžiui, kad pakeistume lygčių sistemos dešines puses, užtenka pakeisti B vektoriaus narius. Ankstesniame žingsnyje minėta matrica yra sudaroma iš pradinių duomenų ir naudojama simplekso algoritme.
4. Kiekvieno ciklo metu pereinama per paskutinės eilutės pirmus N stulpelių ieškant neigiamų reikšmių. Rasta pirma reikšmė nulemia stulpelį, skirtą įtraukti į bazę. Neradus nei vienos neigiamos reikšmės skaičiavimas nutraukiamas ir atspausdinami rezultatai.



Pav. 1 - ABAKS skaičiavimų metu keičiamos matricos vizualus paaiškinimas. Pavaizduota matrica sutampa su pradine matrica, kuri sudaroma sprendžiant optimizavimo uždavinį su pakeistomis dešinėmis lygybių pusėmis.

Detaliau apie atraminės bazės keitimo procesą

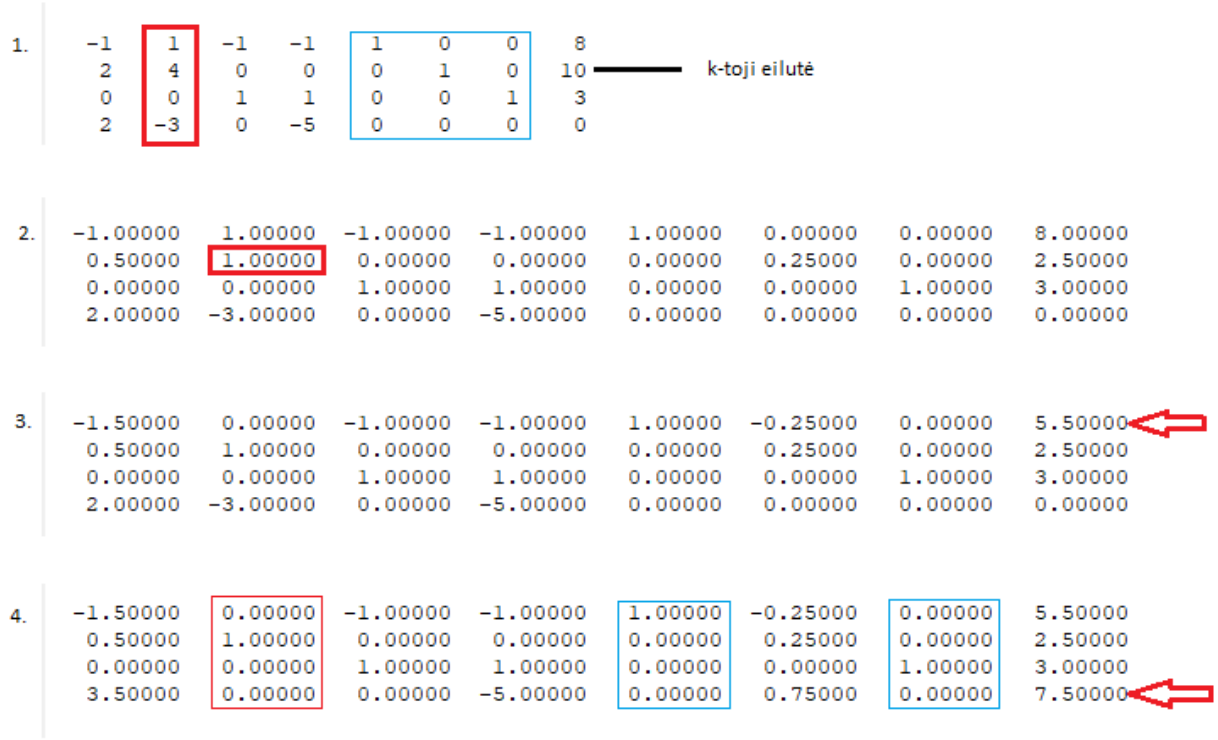
ABAKS'ui nauja bazė parenkama pagrindinėje programoje pereinant per X komponentų koeficientus matricos paskutinėje eilutėje. Kai yra randamas neigiamas koeficientas, yra paimamas to koeficiento stulpelio indeksas ir paduodamas **„abaks"** funkcijai.

Dėl pirmosios bazės sudarymo. Pirmoji bazė sudaryta pasiėmus uždavinio formuluotę matricoje standartiniu pavidalu ir nurodžius kaip bazę tuos stulpelius, kuriuose yra fiktyviųjų kintamųjų diagonalė.

ABAKS funkcijos veikimo žingsniai:

1. **Apskaičiuoti bazinio sprendinio koeficientus** (x_{i0}). Jie gaunami išsprendžiant tokią lygčių sistemą: $X_i = A_b \setminus B$, kur A_b yra matricos poaibis, į kurį įtraukti tik baziniai stulpeliai (tipiškai su vienetine diagonale), o X_i vektoriaus ilgis – bazės stulpelių kiekis.
2. **Nustatyti, kurį stulpelį reikia pašalinti iš bazės.** Tai atliekama skaičiuojant lambda reikšmę, naudojant x_{i0} reikšmes ir einamojo stulpelio x_{ij} reikšmes. Nustačius minimalią lambda reikšmę yra išsaugomas sąlyginis indeksas stulpelio (**k**), kur ji gauta.

3. **Atlikti bazės keitimą.** Šis žingsnis susideda iš dviejų smulkesnių žingsnių:
    a. Padalinti k-tąją matricos eilutę, kad jos elementas naujame baziniame stulpelyje būtų lygus 1.
    b. Atlikti veiksmus su matrica, kad likę naujo bazinio stulpelio elementai taptų nuliais. Tai padaroma pridedant k-tąją eilutę, padaugintą iš realiojo skaičiaus, prie likusių matricos eilučių.



Pav. 2 - ABAKS proceso paaiškinimas su pavyzdžiu. Į bazę (5, 6, 7) įtraukiamas antras stulpelis. **Pirmoje** matricoje atlikti pirmi du žingsniai ir nustatyta, kad reikia pašalinti antrąjį bazės stulpelį. **Antroji** matrica - antroji eilutė padalinta iš 4, kad diagonalėje atsirastų 1. **Trečioji** ir **ketvirtoji** matricos – rezultatai kituose stulpeliuose gavus nulius. Iš pirmosios eilutės buvo atimta antroji, o prie ketvirtosios pridėta antroji, padauginta iš 3. Trečioji matricos eilutė nekeista – ten buvo nulis.
Raudonai – nauja bazės eilutė ir jos pagrindinis elementas. **Mėlynai** – senoji bazė, jos likę stulpeliai.

Pastaba: pagal dabartinį įgyvendinimą bazės keitimas negali būti sėkmingas, jei naujame bazės stulpelyje yra 0 ties pagrindiniu elementu (pavyzdyje – antra eilutė, antras stulpelis). Taip yra dėl to, kad bazės keitimo metu neįmanoma eilutės taip padalinti ar padauginti, kad iš 0 atsirastų 1. Tokiu atveju, būtų galima pirmiau pridėti prie šios eilutės kitą, kurioje nebūtų 0 tame stulpelyje, ir tęsti algoritmą.

Rezultatai

Pirmojo uždavinio rezultatai:

Pirmasis uždavinys yra su sąlygoje nurodytomis dešiniosiomis lygčių pusėmis (B): 8, 10, 3.

1. Apskaičiuota minimumo reikšmė: -22.5,
2. Bazinis sprendinys: (0, 2, 0, 3, 8.5, 0, 0),
3. Bazė: {2, 4, 5}.

Antrojo uždavinio rezultatai:

Antrojo uždavinio dešiniosios lygčių pusės gautos iš studento ID paskutinių trijų skaičių: 2, 3, 9.

1. Apskaičiuota minimumo reikšmė: -47.25,
2. Bazinis sprendinys: (0, 0.75, 0, 9, 10.25, 0, 0),
3. Bazė: {2, 4, 5}.

Galima pastebėti, kad abiem atvejais sprendinys turi tą pačią bazę. Pakeitus dešines lygčių puses pasikeičia tik tikslo funkcijos reikšmė ir bazinio sprendinio komponentų reikšmės. Pirmoji pasikeičia dėl to, kad yra atliekamos eilučių tiesinės transformacijos ir prie tikslo funkcijos reikšmės yra pridedamos dešinės lygčių pusės, padaugintos iš realiųjų skaičių. Antrosios pasikeičia dėl to, kad yra gaunamos iš dešiniųjų lygčių pusių (tiksliau, jos yra lygios dešiniosioms pusėms).

Baziniai sprendiniai patikrinti atskirai, raštu. Jie tenkina apribojimus ir yra leistini – sudaryti iš neneigiamų skaičių. Testavimo metu patikrinta, kad rasti sprendiniai yra tikrai minimumo taškai.

Galutinių matricų palyginimas

-1.50000	0.00000	0.00000	0.00000	1.00000	-0.25000	1.00000	8.50000
0.50000	1.00000	0.00000	0.00000	0.00000	0.25000	0.00000	2.50000
0.00000	0.00000	1.00000	1.00000	0.00000	0.00000	1.00000	3.00000
3.50000	0.00000	5.00000	0.00000	0.00000	0.75000	5.00000	22.50000

-1.50000	0.00000	0.00000	0.00000	1.00000	-0.25000	1.00000	10.25000
0.50000	1.00000	0.00000	0.00000	0.00000	0.25000	0.00000	0.75000
0.00000	0.00000	1.00000	1.00000	0.00000	0.00000	1.00000	9.00000
3.50000	0.00000	5.00000	0.00000	0.00000	0.75000	5.00000	47.25000

Pav. 3 galutinių matricų palyginimas. Viršuje pirmo uždavinio galutinė matrica, apačioje - antro. Raudonai pažymėtos vietos, kuriose skiriasi skaičiai, visur kitur jie sutampa. Galima pastebėti kiekvienoje matricoje po 4 stulpelius, sudarytus iš vieneto ir nulių. Tačiau trys iš jų yra baziniai, nes apatiniai langeliai lygūs nuliui.

Išvados

1. Pavyko įgyvendinti Simplekso algoritmą taip, kad būtų patogu pakeisti apribojimų matricą ir pakeistą uždavinį pavyktų išspręsti.
2. Gauti sprendiniai yra leistini ir tenkina apribojimus.
3. Keičiant apribojimų dešines lygčių puses kinta randamas minimumo taškas. Minimumo taško bazė nekinta.

Priedas – kodo fragmentai

```
function retval = abaks (AB, Base, new_column)
  [rows cols] = size(AB);
  B_index = cols;
  # Step 1: calculate base solution coeficients xi0
  Xi = AB(:, Base) \ AB(:, B_index);

  # Step 2: calculate Lambda (select which new_column to remove from the base)
  counter = 1;
  k = 1;
  min_lambda = inf;
  for i = 1:size(Base)(2)
    xij = AB(i, new_column);
    if xij > 0
      lambda = Xi(i) / xij;
    else
      lambda = inf;
    endif

    if min_lambda > lambda
      min_lambda = lambda;
      k = counter;
    endif
    counter = counter + 1;
  endfor

  # Step 3: calculate new base solution matrix
  # The new new_column should be made of a single 1 and many 0.
  division_coef = AB(k, new_column);
  AB(k, :) /= division_coef;
  rows_v = 1:rows;
  rows_to_edit = rows_v(rows_v != k);
  for row = rows_to_edit
    if AB(row, new_column) != 0
      division_coef = AB(k, new_column) / AB(row, new_column);
      AB(row, :) -= AB(k, :) / division_coef;
    endif
  endfor

  # Some information regarding the result of this function:
  remove_col = Base(k);
  Base(k) = new_column;
  new_base = sort(Base);
  old_base = Base;

  retval = AB;
endfunction
```

Pav. 4 Atraminių bazių sukeitimą įgyvendinanti funkcija.

```
# Setup: variables and functions.
# Goal function:
f = @(x1, x2, x3, x4) 2*x1 - 3*x2 - 5*x4;

# Optimisation problem's restrictions in the standard form:
A = [
  -1  1 -1 -1  1  0  0;
   2  4  0  0  0  1  0;
   0  0  1  1  0  0  1;
];

# The last row will show whether it is worth to insert a column to the base
# Last row, last column - goal function value.
last_row = [
   2 -3  0 -5  0  0  0;
];

B =  [8; 10; 3];
my_B0 = [2; 3; 9];
B = B0;

# Forming the matrix which will be used for the Simplex method:
# Note the zero at the end of the last row. That's the value of the goal functic
AB = [A B];
ABX = [
  AB;
  last_row, 0
];

# Setting some initial Simplex method variables:
ABXi = ABX
[rows cols] = size(ABXi);
my_base = [5, 6, 7];
base = get_base(ABXi); #my_base;
changed = true;
iter_count = 0;
```

Pav. 5 pagrindinės programos pirmoji dalis - pradinių duomenų aprašymas ir matricos konstravimas.

```
# Main code. Simplex method cycle:
while changed == true #&& iter_count < 60
  changed = false;
  for i = 1:(cols - 1)
    if ABXi(rows, i) < 0
      ABXi = abaks(ABXi, base, i)
      base = get_base(ABXi);
      changed = true;
      break;
    endif
  endfor
  iter_count += 1;
endwhile

# Results section:
printf("\n\n ---------RESULTS:------------ \n")
# Calculating the base solution:
AB = ABXi(1:(rows-1), :);
base_solution_xi = AB(:, base) \ AB(:, cols);
X_minimum = [];
i = 1;
for j = 1:(cols-1)
  if i <= size(base)(2) && j == base(i)
    X_minimum = [X_minimum base_solution_xi(i)];
    i += 1;
  else
    X_minimum = [X_minimum 0];
  endif
endfor

# Printing the results:
X_minimum
minimum_value = ABXi(rows, cols) * (-1)
base
```

Pav. 6 pagrindinės programos antroji pusė - Simplekso algoritmas ir rezultatų spausdinimas.

```
function base = get_base (matrix)
  base_cols = [];
  [rows cols] = size(matrix);
  for col = 1:(cols - 1)
    ones = 0;
    for row = 1:rows
      cell = matrix(row, col);
      if (cell == 1) || (cell == 0)
        ones += cell;
      else
        ones = -1;
        break;
      endif
    endfor
    if ones == 1
      base_cols = [base_cols col];
    endif
  endfor

  base = base_cols;
endfunction
```

Pav. 7 pagalbinė funkcija, kuri grąžina matricos bazinių stulpelių indeksus, jei tokių stulpelių yra.