VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

Tiesinis programavimas

Ketvirto laboratorinio darbo ataskaita

Atliko: Paulina Podgaiska

VU el. p.: paulina.podgaiska@mif.stud.vu.lt

Vertino: Vyr. M. Darbuot., Dr. (HP), Julius Žilinskas

Vilnius
2025

UŽDUOTIS

Ketvirto laboratorinio darbo užduotis pateikta lentelėje 1

1lentelė. Ketvirto laboratorinio darbo užduotis

Tiesinis programavimas
1. Suprogramuokite simplekso algoritmą tiesinio programavimo uždaviniams.
2. Užrašykite duotą uždavinį matriciniu pavidalu standartine forma.
3. Išspręskite uždavinį suprogramuotu simplekso algoritmu.
4. Pakeiskite apribojimų dešinės pusės konstantas į a, b ir c – studento knygelės numerio “xxxxabc” skaitmenis. Išspręskite individualų uždavinį suprogramuotu simplekso algoritmu.
5. Palyginkite uždavinių sprendimo rezultatus: minimali tikslo funkcijos reikšmė, optimalus sprendinys ir bazė.

ATASKAITA

1. DARBO EIGA

Tiesinio programavimo užduotis – minimizuoti tikslo funkciją:

$$f(X) = 2x_1 - 3x_2 - 5x_4,$$

turint apribojimus:

$$-x_1 + x_2 - x_3 - x_4 \leq 8$$
$$2x_1 + 4x_2 \leq 10$$
$$x_3 + x_4 \leq 3$$
$$x_i \geq 0,$$

čia 8, 10, 3 yra apribojimų dešinės pusės konstantos, kurios vėliau bus pakeistos pagal studento knygelės numerio skaitmenis a=9, b=9, c=9.

Gauti rezultatai bei programos kodas, atliktas naudojant „Python“ programavimo kalbą, pateikti žemiau (programos kodas pateiktas priede).

1.1. Simplekso algoritmas

Simplekso algoritmas randa optimalų sprendinį pereinant nuo vienos bazės prie kitos, kiekvieną kartą gerinant tikslo funkcijos reikšmę. Kad būtų galima naudoti šį metodą, uždavinį perrašome standartine forma:

$$\min c^T x$$
$$Ax = b, x \geq 0$$

Perrašant uždavinį standartine forma, uždavinio kintamųjų skaičius padidėja tiek, kiek bendrajame uždavinyje yra nelygyninių apribojimų, todėl atsiranda slankieji kintamieji s_1, s_2, s_3 ir nelygybės virsta lygybėmis:

$$-x_1 + x_2 - x_3 - x_4 + s_1 = 8$$
$$2x_1 + 4x_2 + s_2 = 10$$
$$x_3 + x_4 + s_3 = 3$$

Iš koeficientų sudaroma A matrica:

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

, o tikslo funkcijos vektorius (c):

$$c = [2, -3{,}0, -5{,}0{,}0{,}0]$$

Naudojami slankieji kintamieji s_1, s_2, s_3 kaip pradiniai baziniai kintamieji, nes jų koeficientai sudaro vienetinę matricą. Apskaičiuojame bazinių kintamųjų reikšmes sprendžiant:

$$x_B = B^{-1}b$$

kur B – bazinių stulpelių matrica, b – apribojimų dešinė pusė.

Toliau skaičiuojamas vektorius:

$$z = c_B^T B^{-1} A, \qquad ir\ reduced\_costs = c - z$$

Jei visi reduced_costs yra didesni arba lygūs nuliui, tai sprendimas yra optimalus. Įeinantis kintamasis yra pasirinktas kaip tas, kuris turi mažiausią reduced_cost, nes jis greičiausiai pagerins tikslo funkciją. Išeinantis kintamasis pasirenkamas užtikrinus jo neneigiamumą, taigi apskaičiuojame kryptį $d = B^{-1}a_j$:

$$\min_i(\frac{x_{b_i}}{d_i}), \qquad kai\ d_i > 0$$

Naujas kintamasis įtraukiamas į bazę, senasis – pašalinamas. Šie žingsniai kartojami tol, kol visi reduced_costs tampa teigiami arba nuliai, kas reiškia, kad pasiektas optimalus sprendimas. Demonstruojama pirma iteracija su pradinėmis dešiniosiomis konstantomis:

Sudaroma bazės matrica B

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Atvirkštinė B matrica B^{-1} sudauginama su $b = [8,10,3]$ ir gauname bazini sprendinį $x_B = [8,10,3]$.Visi nebaziniai kintamieji virsta nuliais ir užrašome x vektorių - [0,0,0,0,8,10,3]. Toliau pagal bazės indeksus yra gaunamas c_B^T vektorius $c_B^T = [0,0,0]$ ir skaičiuojamas $z = [0,0,0,0,0,0,0]$. Skaičiuojamas $reduced\_costs = c - z = c$. Toliau atrenkamas įeinantis kintamasis, kurio reikšmė yra mažiausia, šiuo atveju tai yra -5, o tai yra x_4. Išeinantis kintamasis randamas apskaičiuojant d kryptį, $d = [-1,0,1]$. Toliau $\frac{x_B}{d}$, $\frac{8}{1} = 8$, $\frac{10}{0} = inf, \frac{3}{1} = 3$, mažiausia reikšmė yra 3, todėl atitinkamai išeina s_3. Atnaujinta bazė yra s_1, s_2, x_4. Kartojamas tas pats procesas, kol nelieka neigiamų reikšmių reduced_costs.

1.2. Rezultatai

Išspręsti uždaviniai simplekso algoritmo pagalba yra palyginami lentelėje, taip pat parengta atskira sprendinių bazinių kintamųjų lentelė. Su uždavinio sąlygos dešiniosiomis konstantomis, optimalus sprendinys yra:

$$\begin{cases} x_1 = 0 \\ x_2 = 2{,}5 \\ x_3 = 0 \\ x_4 = 3 \\ s_1 = 8{,}5 \\ s_2 = 0 \\ s_3 = 0 \end{cases}$$

Patikrinta, kad lygybės yra tenkinamos:

$$\begin{cases} -0 + 2{,}5 - 0 - 3 + 8{,}5 = 8 \\ 2 * 0 + 4 * 2{,}5 + 0 = 10 \\ 0 + 3 + 0 = 3 \end{cases}$$

Visi kintamieji yra teigiami, todėl sprendinys yra leistinas. Sprendinio funkcijos reikšmė yra $f(X)_{min} = 2 * 0 - 3 * 2{,}5 - 5 * 3 = -22{,}5$.

Optimalus sprendinys su individualiomis dešiniosiomis konstantomis 9, 9, 9:

$$\begin{cases} x_1 = 0 \\ x_2 = 2{,}25 \\ x_3 = 0 \\ x_4 = 9 \\ s_1 = 15{,}75 \\ s_2 = 0 \\ s_3 = 0 \end{cases}$$

Patikrinimas ar lygybės yra tenkinamos:

$$\begin{cases} -0 + 2{,}25 - 0 - 9 + 15{,}75 = 9 \\ 2 * 0 + 4 * 2{,}25 + 0 = 9 \\ 0 + 9 + 0 = 9 \end{cases}$$

Visi kintamieji yra teigiami. Sprendinio funkcijos reikšmė yra $f(X)_{min} = 2 * 0 - 3 * 2{,}25 - 5 * 9 = -51{,}75$.

Sprendinių palyginimas matomas 1 pav. Matome, kad keičiant dešiniąsias puses, optimalus sprendinys pasikeičia, o tikslo funkcijos reikšmė akivaizdžiai sumažėja.

=== Sprendinių palyginimas ===

Versija	x1	x2	x3	x4	s1	s2	s3	f(X) (min)
Pradinis uzdavinys	0	2.5	0	3	8.5	0	0	-22.5
Individualus (9, 9, 9)	0	2.25	0	9	15.75	0	0	-51.75

1 pav. Optimalių sprendinių ir minimalių tikslo funkcijos reikšmių palyginimo lentelė

Taip pat paruošta sprendinių bazių palyginimo lentelė 2 pav. Abiem atvejais bazės yra vienodos, nes abu vektoriai yra tame pačiame leidžiamajame daugiakampyje ir optimalus sprendinys atitinka tą pačią bazinę sritį.

=== Bazės palyginimas ===

Versija	B1	B2	B3
Pradinis uzdavinys	s1	x2	x4
Individualus (9, 9, 9)	s1	x2	x4

2 pav. Rastų sprendinių bazių palyginimas

2. IŠVADOS

Išspręstas tiesinio programavimo uždavinys taikant simplekso algoritmą. Pradinės apribojimų dešiniosios pusės reikšmės (8, 10, 3) ir individualios reikšmės (9, 9, 9) lėmė skirtingus optimalius sprendinius ir tikslo funkcijos reikšmes, tačiau baziniai kintamieji abiejose situacijose išliko tie patys. Abiem atvejais sprendiniai tenkina visas lygybes ir visų kintamųjų reikšmės yra neneigiamos ($x_i \geq 0$), todėl sprendiniai yra leistini.

3. PRIEDAS

Žemiau pateikiamas kodas su anksčiau aprašytu įgyvendintu algoritmu.

4_laboratorinis.py

```
import numpy as np
from tabulate import tabulate

c_vector = np.array([2, -3, 0, -5, 0, 0, 0], dtype=float)

A = np.array([
    [-1,  1, -1, -1, 1, 0, 0],
    [ 2,  4,  0,  0, 0, 1, 0],
    [ 0,  0,  1,  1, 0, 0, 1]
], dtype=float)

b_salyga = np.array([8, 10, 3], dtype=float)
b_individual = np.array([9, 9, 9], dtype=float)

def simpleksas(c, A, b, baziniai):
    m, n = A.shape
    baziniai = baziniai.copy()
    while True:
        B = A[:, baziniai]
        c_b = c[baziniai]
        B_atvirkst = np.linalg.inv(B)
        x_b = B_atvirkst @ b
        x = np.zeros(n)
        for i, idx in enumerate(baziniai):
            x[idx] = x_b[i]
        z = c_b @ B_atvirkst @ A
        reduced_costs = c - z
        if all(reduced_costs >= -1e-8):
            break
        ieinantis = np.argmin(reduced_costs)
        kryptis_d = B_atvirkst @ A[:, ieinantis]
        iseinantis = [x_b[i] / kryptis_d[i] if kryptis_d[i] > 1e-8 else np.inf for i
in range(len(kryptis_d))]
        iseinancio_indeksas = np.argmin(iseinantis)
        if iseinantis[iseinancio_indeksas] == np.inf:
            break
        baziniai[iseinancio_indeksas] = ieinantis
    return x, c @ x, baziniai

def sprendimas(A, b, c):
    pradine_baze = [4, 5, 6]
    sol_x, z, baze = simpleksas(c, A, b, pradine_baze)
    sprend_kintam = sol_x[:4]
    si_kintam = sol_x[4:]
    return sprend_kintam, si_kintam, z, baze

sol_x, sol_s, z_sprend, baze = sprendimas(A, b_salyga, c_vector)
sol_x_indiv, sol_s_indiv, z_indiv, baze_indiv = sprendimas(A, b_individual, c_vector)

# === lentele palyg
antraste = [f"x{i+1}" for i in range(4)] + ["s1", "s2", "s3"]
sprendiniai = [
    ["Pradinis uzdavinys"] + list(np.round(sol_x, 3)) + list(np.round(sol_s, 3)) +
[round(z_sprend, 3)],
    ["Individualus (9, 9, 9)"] + list(np.round(sol_x_indiv, 3)) +
list(np.round(sol_s_indiv, 3)) + [round(z_indiv, 3)]
]
```

```
print("\n=== Sprendinių palyginimas ===")
print(tabulate(sprendiniai, headers=["Versija"] + antraste + ["f(X) (min)"],
tablefmt="fancy_grid"))

# === baziu lentele
vardai = [f"x{i+1}" for i in range(4)] + ["s1", "s2", "s3"]
baziu_lentele = [
    ["Pradinis uzdavinys"] + [vardai[i] for i in baze],
    ["Individualus (9, 9, 9)"] + [vardai[i] for i in baze_indiv]
]
kiek_baziu = max(len(baze), len(baze_indiv))
bazes_antraste = ["Versija"] + [f"B{i+1}" for i in range(kiek_baziu)]

print("\n=== Bazės palyginimas ===")
print(tabulate(baziu_lentele, headers=bazes_antraste, tablefmt="fancy_grid"))
```