МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

Кафедра информатики, математического и компьютерного моделирования

ОТЧЕТ

по лабораторной работе
по дисциплине «Методы оптимизации»

Выполнил студент
гр. Б9119-02.03.01сцт
Панченко Н.К. _____
(ФИО) *(подпись)*

«17» мая 2022 г.

г. Владивосток
2022

Постановка задачи

Пусть даны прямая и двойственные задачи:

$$\begin{cases} c \cdot x \to \max \\ A \cdot x \leq b \\ x \geq 0 \end{cases} \qquad (1)$$

$$\begin{cases} b \cdot y \to \min \\ A^T \cdot y \geq c \\ y \geq 0 \end{cases} \qquad (2)$$

Необходимо, используя симплекс-метод, найти оптимальное решение задач прямой и двойственной.

Все данные будем генерировать случайным образом и будем перестраивать полученную симплексную таблицу до тех пор, пока все значения из индексной строки не будут положительными. На основе последней таблицы также получим решение двойственной задачи. Также проверим, что $c \cdot x = b \cdot y$.

Реализация алгоритма

Для реализации воспользуемся языком программирования Python и библиотекой Numpy для удобной работы с матрицами.

```
import numpy as np
import copy

n = 6
m = 8

A = np.random.randint(1, 10, (m, n)).astype('float')
b = np.random.randint(1, 10, m).astype('float')
```

```
c = np.random.randint(1, 10, n).astype('float')

startC = copy.deepcopy(c)
startB = copy.deepcopy(b)
prevLeads = []
print("A =")
print(A)
print("b = " + str(b))
print("c = " + str(c))
c = c*(-1)
cFree = 0

colInd = [i for i in range(0, n)]
prevLeads.append(copy.deepcopy(colInd))
strInd = [i for i in range(n, n+m)]

while min(c) < 0:

    oldColInd = copy.deepcopy(colInd)
    oldStrInd = copy.deepcopy(strInd)

    changeC = copy.deepcopy(c)
    check = True

    while check:
        check = False
        leadCol = np.argmax(np.abs(changeC))
        leadStr = 0
        leadStrVal = 100000000000000000

        for i in range(0, m):
            if (not((b[i] > 0) and (A[i, leadCol] < 0))):
                leadStrNew = b[i] / A[i, leadCol]
                if (leadStrNew < leadStrVal):
                    leadStr = i
                    leadStrVal = leadStrNew
```

```
        leadVal = copy.deepcopy(A[leadStr, leadCol])


        strInd = np.insert(strInd, 0, colInd[leadCol])
        colInd[leadCol] = copy.deepcopy(strInd[leadStr+1])
        strInd = np.delete(strInd, leadStr + 1)


        for i in range(len(prevLeads)):
            prevLeads[i].sort()
            sortColInd = copy.deepcopy(colInd)
            sortColInd.sort()
            if prevLeads[i] == sortColInd:
                changeC[leadCol] = 0
                colInd = copy.deepcopy(oldColInd)
                strInd = copy.deepcopy(oldStrInd)
                check = True
                break

    prevLeads.append(copy.deepcopy(colInd))


    helpVals = copy.deepcopy(A[:, leadCol]*(-1))
    helpVals = np.delete(helpVals, leadStr)


    leadStrVals = A[leadStr]
    leadB = b[leadStr]

    A = np.delete(A, leadStr, 0)
    b = np.delete(b, leadStr)
    A = np.reshape(np.insert(A, 0, [0 for i in range(0, n)]), (m, n))
    b = np.insert(b, 0, 0)

    for i in range(0, n):
        if i != leadCol:
            A[0, i] = leadStrVals[i]/leadVal
    A[0, leadCol] = 1/leadVal
```

```
    b[0] = leadB/leadVal


    for i in range(1, m):
        for j in range(0, n):
            oldVal = A[i, j]
            A[i, j] = A[0, j]*helpVals[i-1]
            if (oldColInd[j] == colInd[j]):
                A[i, j] += oldVal
        b[i] = b[0]*helpVals[i-1] + b[i]


    leadC = c[leadCol]


    for i in range(0, n):
        oldVal = c[i]
        c[i] = A[0, i] * leadC *(-1)
        if (oldColInd[i] == colInd[i]):
            c[i] += oldVal
    cFree = b[0] * leadC*(-1) + cFree

print()


res = 0
for i in range(0, n):
    if i in strInd:
        print("x_" + str(i) + " = " + str(b[list(strInd).index(i)]) +
            " ", end='')
    else:
        print("x_" + str(i) + " = " + str(0) + " ", end='')

print()
check = False
for i in range(0, n):
    if i in strInd:
```

```
        if check:
            print(" + ", end='')
        print(str(startC[i]) +" * " + str(b[list(strInd).index(i)]),
            end='')
        res += startC[i]*b[list(strInd).index(i)]
        check = True


print(" = " + str(res))

res = 0

for i in range(n, n+m):
    if i in colInd:
        print("y_" + str(i-n) + " = " + str(c[list(colInd).index(i)])
            + " ", end='')
    else:
        print("y_" + str(i-n) + " = " + str(0) + " ", end='')

print()

check = False
for i in range(n, n+m):
    if i in colInd:
        if check:
            print(" + ", end='')
        print(str(startB[i-n]) +" * " +
            str(c[list(colInd).index(i)]), end='')
        res += startB[i-n]*c[list(colInd).index(i)]
        check = True

print(" = " + str(res))
```

Тесты

```
A =
[[2. 9. 1. 2. 5. 3.]
 [7. 8. 1. 2. 1. 2.]
 [3. 5. 8. 8. 8. 3.]
 [6. 7. 3. 3. 2. 8.]
 [4. 2. 4. 5. 2. 3.]
 [1. 7. 1. 2. 4. 9.]
 [5. 1. 7. 1. 5. 5.]
 [4. 3. 4. 9. 9. 5.]]
b = [4. 4. 9. 5. 9. 4. 7. 4.]
c = [9. 5. 7. 5. 8. 7.]

Решение прямой задачи:
x_0 = 0.5714285714285714 x_1 = 0 x_2 = 0 x_3 = 0 x_4 = 0 x_5 = 0
9.0 * 0.5714285714285714 = 5.142857142857142
Решение обратной задачи(проверка):
y_0 = 0 y_1 = -7.649873777082677e-05 y_2 = 0 y_3 = 0 y_4 = 0 y_5 = 0 y_6 = 0 y_7 = 0
4.0 * -7.649873777082677e-05 = -0.0003059949510833071
```

Заключение

В ходе данной лабораторной работы был изучен симплекс-метод, а также разработана программа, позволяющая находить решения прямой и двойственных задач линейной оптимизации симплекс-методом.