МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

Лабораторная работа № 1
Реализация симплекс метода для решения задач линейного программирования

По дисциплине «Прикладная математика»

Выполнили студенты групп
M32011
Лунев Илья Андреевич
Семенов Георгий Витальевич
Смирнов Сергей Викторович

Преподаватель:
Москаленко Мария Александровна

САНКТ-ПЕТЕРБУРГ

2021

I. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1. **Общая и** каноническая форма задач линейного программирования.

Общей задачей линейного программирования называется задача, которая состоит в определении максимального (минимального) значения функции

$$F(x) = \sum_{j=1}^{n} c_j x_j \; (1.0), \qquad \text{при условиях}$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, i = \overline{1, k}; k \le m, \; (1.1)$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, i = \overline{k+1, m}, (1.2)$$

$$x_j \ge 0, j = \overline{1, s}; s \le n \; (1.3)$$

Функция (1.1) называется **целевой функцией** (или **линейной формой**) задачи (1.0) – (1.3), а условия (1.0) – (1.3) – ограничениями данной задачи. **Канонической** (или **основной**) **задачей линейного программирования** называется задача, которая состоит в определении максимального (минимального) значения функции (1.0) при выполнении условий (1.0) и (1.3), где $k = 0, s = n$.

2. **Приведение задачи к каноническому виду.**

Для приведения задачи линейного программирования к каноническому виду необходимо выполнить следующие преобразования:

- Задачу максимизации привести к задаче минимизации путем умножения функции на -1
- Ограничения виды ≥ привести к ограничениям вида ≤ путем умножения неравенств на -1
- Привести неравенства к равенствам путём введения новых переменных

За выполнение данных преобразований отвечает функция:

```
    def to_canonical(self):
```

II. РЕЗУЛЬТАТЫ РАБОТЫ АЛГОРИТМА

1.

$$f(x) = -6x_1 - x_2 - 4x_3 + 5x_4 \to min,$$

$$\begin{cases} 3x_1 + x_2 - x_3 + x_4 = 4, \\ 5x_1 + x_2 + x_3 - x_4 = 4, \\ x_j \geq 0, \quad j = 1, 2, 3, 4. \end{cases}$$

В качестве исходного базисного допустимого решения можно взять точку: $\bar{x} = (1, 0, 0, 1)^T$.

2.

$$f(x) = -x_1 - 2x_2 - 3x_3 + x_4 \to min,$$

$$\begin{cases} x_1 - 3x_2 - x_3 - 2x_4 = -4, \\ x_1 - x_2 + x_3 = 0, \\ x_j \geq 0, \quad j = 1, 2, 3, 4. \end{cases}$$

В качестве исходного базисного допустимого решения можно взять точку: $\bar{x} = (0, 1, 1, 0)^T$.

3.

$$f(x) = -x_1 - 2x_2 - x_3 + 3x_4 - x_5 \to min,$$

$$\begin{cases} x_1 + x_2 + 2x_4 + x_5 = 5, \\ x_1 + x_2 + x_3 + 3x_4 + 2x_5 = 9, \\ x_2 + x_3 + 2x_4 + x_5 = 6x_j \geq 0, \quad j = 1, 2, 3, 4, 5. \end{cases}$$

В качестве исходного базисного допустимого решения можно взять точку: $\bar{x} = (0, 0, 1, 2, 1)^T$.

4.

$$f(x) = -x_1 - x_2 - x_3 + x_4 - x_5 \to min,$$

$$\begin{cases} x_1 + x_2 + 2x3 = 4, \\ -2x_2 - 2x_3 + x_4 - x_5 = -6, \\ x_1 - x_2 + 6x_3 + x_4 + x_5 = 12x_j \geq 0, \quad j = 1, 2, 3, 4, 5. \end{cases}$$

В качестве исходного базисного допустимого решения можно взять точку: $\bar{x} = (1, 1, 2, 0, 0)^T$.

5.

$$f(x) = -x_1 + 4x_2 - 3x_3 + 10x_4 \to min,$$

$$\begin{cases} x_1 + x_2 - x_3 - 10x_4 = 0, \\ x_1 + 14x_2 + 10x_3 - 10x_4 = 11, \\ x_j \geq 0, \quad j = 1, 2, 3, 4. \end{cases}$$

6.

$$f(x) = -x_1 + 5x_2 + x_3 - x_4 \to min,$$

$$\begin{cases} x_1 + 3x_2 + 3x_3 + x_4 \le 3, \\ 2x_1 + 3x_3 - x_4 \le 4, \\ x_j \ge 0, \quad j = 1, 2, 3, 4. \end{cases}$$

.

7.

$$f(x) = -x_1 - x_2 + x_3 - x_4 + 2x_5 \to min,$$

$$\begin{cases} 3x_1 + x_2 + x_3 + x_4 - 2x_5 = 10, \\ 6x_1 + x_2 + 2x_3 + 3x_4 - 4x_5 = 20, \\ 10x_1 + x_2 + 3x_3 + 6x_4 - 7x_5 = 30, \\ x_j \ge 0, \quad j = 1, 2, 3, 4, 5. \end{cases}$$

```
Now using initial feasible solutions, if available.
+  [0. 5.]  OK  tasks/example.json
+  [3. 2.]  OK  tasks/example2.json
+  [6. 4.]  OK  tasks/example3.json
+  [0. 4. 0. 0.]  OK  tasks/t1.json
+  [2. 2. 0. 0.]  OK  tasks/t2.json
+  [3. 2. 4. 0. 0.]  OK  tasks/t3.json
+  [4. 0. 0. 1. 7.]  OK  tasks/t4.json
+  [1. 0. 1. 0.]  OK  tasks/t5.json
+  [2.33333333 0.         0.         0.66666667]  OK  tasks/t6.json
+  [ 0.  0. 10.  0.  0.]  OK  tasks/t7.json

Now solving with supplementary task.
+  [0. 5.]  OK  tasks/example.json
+  [3. 2.]  OK  tasks/example2.json
+  [6. 4.]  OK  tasks/example3.json
+  [0. 4. 0. 0.]  OK  tasks/t1.json
+  [2. 2. 0. 0.]  OK  tasks/t2.json
+  [3. 2. 4. 0. 0.]  OK  tasks/t3.json
+  [4. 0. 0. 1. 7.]  OK  tasks/t4.json
+  [1. 0. 1. 0.]  OK  tasks/t5.json
+  [2.33333333 0.         0.         0.66666667]  OK  tasks/t6.json
+  [ 0.  0. 10.  0.  0.]  OK  tasks/t7.json
```

Подробное решение первого примера (debug=True)

```
Now using initial feasible solutions, if available.
[[ 4.  3.  1. -1.  1.]
 [ 4.  5.  1.  1. -1.]
```

```
 [ 4.  6.  1.  4. -5.]]
Rows  [0 3]
[[ 1.33333333  1.          0.33333333 -0.33333333  0.33333333]
 [-2.66666667  0.         -0.66666667  2.66666667 -2.66666667]
 [-4.          0.         -1.          6.         -7.        ]]
Rows  [0 3]
[[ 1.    1.    0.25  0.    0.  ]
 [ 1.   -0.    0.25 -1.    1.  ]
 [ 3.    0.    0.75 -1.    0.  ]]
Rows  [0 3]
==================
Next step. Table:
[[ 1.    1.    0.25  0.    0.  ]
 [ 1.   -0.    0.25 -1.    1.  ]
 [ 3.    0.    0.75 -1.    0.  ]]
Point: [1. 0. 0. 1.]
Rows: [0 3]
Pivot Index:  0 ,  2
Pivot:  0.25
==================
Next step. Table:
[[ 4.  4.  1.  0.  0.]
 [ 0. -1.  0. -1.  1.]
 [ 0. -3.  0. -1.  0.]]
Point: [0. 4. 0. 0.]
Rows: [1 3]
Answer found.
+  [0. 4. 0. 0.]  OK  tasks/t1.json
Now solving with supplementary task.
[[ 4.  3.  1. -1.  1.  1.  0.]
 [ 4.  5.  1.  1. -1.  0.  1.]
 [ 8.  8.  2.  0.  0.  0.  0.]]
Rows  [4 5]
[[ 4.  3.  1. -1.  1.  1.  0.]
 [ 4.  5.  1.  1. -1.  0.  1.]
 [ 8.  8.  2.  0.  0.  0.  0.]]
Rows  [4 5]
[[ 4.  3.  1. -1.  1.  1.  0.]
 [ 4.  5.  1.  1. -1.  0.  1.]
 [ 8.  8.  2.  0.  0.  0.  0.]]
Rows  [4 5]
==================
Next step. Table:
[[ 4.  3.  1. -1.  1.  1.  0.]
 [ 4.  5.  1.  1. -1.  0.  1.]
 [ 8.  8.  2.  0.  0.  0.  0.]]
Point: [0. 0. 0. 0. 4. 4.]
Rows: [4 5]
Pivot Index:  1 ,  1
```

```
Pivot:  5.0
==================
Next step. Table:
[[ 1.6  0.   0.4 -1.6  1.6  1.   -0.6]
 [ 0.8  1.   0.2  0.2 -0.2  0.    0.2]
 [ 1.6  0.   0.4 -1.6  1.6  0.   -1.6]]
Point: [0.8 0.  0.  0.  1.6 0. ]
Rows: [4 0]
Pivot Index:  0 ,  4
Pivot:  1.6
==================
Next step. Table:
[[ 1.     0.     0.25  -1.     1.     0.625 -0.375]
 [ 1.     1.     0.25   0.     0.     0.125  0.125]
 [ 0.     0.     0.     0.     0.    -1.    -1.   ]]
Point: [1. 0. 0. 1. 0. 0.]
Rows: [3 0]
Answer found.
[[ 4.  3.  1. -1.  1.]
 [ 4.  5.  1.  1. -1.]
 [ 0.  6.  1.  4. -5.]]
Rows  [3 0]
[[ 4.  3.  1. -1.  1.]
 [ 8.  8.  2.  0.  0.]
 [20. 21.  6. -1.  0.]]
Rows  [3 0]
[[ 1.    0.    0.25 -1.    1.  ]
 [ 1.    1.    0.25  0.    0.  ]
 [-1.    0.    0.75 -1.    0.  ]]
Rows  [3 0]
==================
Next step. Table:
[[ 1.    0.    0.25 -1.    1.  ]
 [ 1.    1.    0.25  0.    0.  ]
 [-1.    0.    0.75 -1.    0.  ]]
Point: [1. 0. 0. 1.]
Rows: [3 0]
Pivot Index:  0 ,  2
Pivot:  0.25
==================
Next step. Table:
[[ 4.  0.  1. -4.  4.]
 [ 0.  1.  0.  1. -1.]
 [-4.  0.  0.  2. -3.]]
Point: [0. 4. 0. 0.]
Rows: [1 0]
Pivot Index:  1 ,  3
Pivot:  1.0
==================
```

```
Next step. Table:
[[ 4.  4.  1.  0.  0.]
 [ 0.  1.  0.  1. -1.]
 [-4. -2.  0.  0. -1.]]
Point: [0. 4. 0. 0.]
Rows: [1 2]
Answer found.
+  [0. 4. 0. 0.]  OK  tasks/t1.json
```

Вывод: таким образом симплекс-метод является универсальным методом, с помощью которого можно решить любую задачу линейного программирования, в отличии от графического метода, пригодного лишь для решения систем с двумя переменными