

**Министерство образования Российской Федерации  
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ  
им. Н.Э. БАУМАНА**

Факультет: Информатика и системы управления  
Кафедра: Информационная безопасность (ИУ8)

**Теория принятия решений в условиях информационных конфликтов**

**Лабораторная работа №3**

“Решение ЗЛП методом ветвей и границ”

Вариант 13

**Преподаватель: Коннова Н. С.  
Студент: Андреев Г.А.  
Группа: ИУ8-71**

2021 г.

**Цель работы** — Изучить постановку задачи целочисленного линейного программирования; получить решение задачи ЦЛП методом ветвей и границ.

## Постановка задачи

Требуется найти решение следующей задачи линейного программирования (ЛП):

Методом ветвей и границ найти среди всех  $n$ -мерных векторов  $x = (x_1, x_2, \dots, x_n)$ , удовлетворяющих системе:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \leq b_1 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \leq b_m \end{cases},$$

Такой, для которого достигается минимум ЦФ:

$$\min F = c_1x_1 + c_2x_2 + \dots + c_nx_n$$

## Условие

$$c = [3, 3, 7]$$

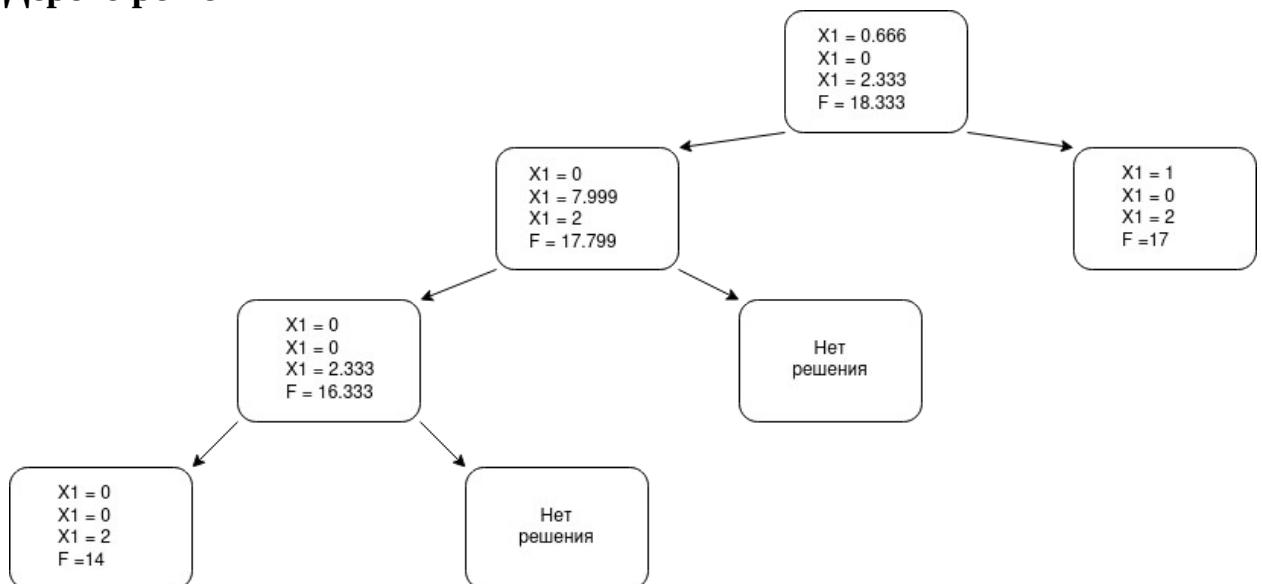
$$b = [3, 5, 7]$$

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 0 \\ 0 & 0.5 & 3 \end{bmatrix}$$

$$[1, 4, 0],$$

$$[0, 0.5, 3]$$

## Дерево решений



## Решение

Метод ветвей и границ:

---

	S0	x_1	x_2	x_3
x_4	3	1	1	1
x_5	5	1	4	0
x_6	7	0	0.5	3
F	0	-3	-3	-7

---

Ищем оптимальное решение:

Индекс разрешающего элемента: (2, 3)

	S0	x_1	x_2	x_6
x_4	0.666667	1	0.833333	-0.333333
x_5	5	1	4	0
x_3	2.33333	0	0.166667	0.333333
F	16.3333	-3	-1.83333	2.33333

---

Индекс разрешающего элемента: (0, 1)

	S0	x_4	x_2	x_6
x_1	0.666667	1	0.833333	-0.333333
x_5	4.33333	-1	3.16667	0.333333
x_3	2.33333	0	0.166667	0.333333
F	18.3333	3	0.666667	1.33333

---

x1 = 0.6666666666666665

x2 = 0

x3 = 2.3333333333333335

F = 18.333333333333332

Ветвимся влево по переменной  $x_0 \leq 0$

	S0	$x_1$	$x_2$	$x_3$
$x_4$	3	1	1	1
$x_5$	5	1	4	0
$x_6$	7	0	0.5	3
$x_7$	0	1	0	0
F	0	-3	-3	-7

Ищем оптимальное решение:

Индекс разрешающего элемента: (2, 3)

	S0	$x_1$	$x_2$	$x_6$
$x_4$	0.666667	1	0.833333	-0.333333
$x_5$	5	1	4	0
$x_3$	2.33333	0	0.166667	0.333333
$x_7$	0	1	0	0
F	16.3333	-3	-1.83333	2.33333

Индекс разрешающего элемента: (3, 1)

	S0	$x_7$	$x_2$	$x_6$
$x_4$	0.666667	-1	0.833333	-0.333333
$x_5$	5	-1	4	0
$x_3$	2.33333	0	0.166667	0.333333
$x_1$	0	1	0	0
F	16.3333	3	-1.83333	2.33333

Индекс разрешающего элемента: (0, 2)

	S0	x_7	x_4	x_6
x_2	0.8	-1.2	1.2	-0.4
x_5	1.8	3.8	-4.8	1.6
x_3	2.2	0.2	-0.2	0.4
x_1	0	1	0	0
F	17.8	0.8	2.2	1.6

$$x_1 = 0.0$$

$$x_2 = 0.7999999999999998$$

$$x_3 = 2.2$$

$$F = 17.799999999999997$$

Ветвимся влево по переменной  $x_1 \leq 0$

	S0	x_1	x_2	x_3
x_4	3	1	1	1
x_5	5	1	4	0
x_6	7	0	0.5	3
x_7	0	1	0	0
x_8	0	0	1	0
F	0	-3	-3	-7

Ищем оптимальное решение:

Индекс разрешающего элемента: (2, 3)

	S0	x_1	x_2	x_6
x_4	0.666667	1	0.833333	-0.333333
x_5	5	1	4	0
x_3	2.33333	0	0.166667	0.333333
x_7	0	1	0	0
x_8	0	0	1	0
F	16.3333	-3	-1.83333	2.33333

Индекс разрешающего элемента: (3, 1)

	S0	x_7	x_2	x_6
x_4	0.666667	-1	0.833333	-0.333333
x_5	5	-1	4	0
x_3	2.33333	0	0.166667	0.333333
x_1	0	1	0	0
x_8	0	0	1	0
F	16.3333	3	-1.83333	2.33333

Индекс разрешающего элемента: (4, 2)

	S0	x_7	x_8	x_6
x_4	0.666667	-1	-0.833333	-0.333333
x_5	5	-1	-4	0
x_3	2.33333	0	-0.166667	0.333333
x_1	0	1	0	0
x_2	0	0	1	0
F	16.3333	3	1.83333	2.33333

x1 = 0.0

x2 = 0.0

x3 = 2.3333333333333335

F = 16.333333333333332

Ветвимся влево по переменной x\_2 <= 2

	S0	x_1	x_2	x_3
x_4	3	1	1	1
x_5	5	1	4	0
x_6	7	0	0.5	3
x_7	0	1	0	0
x_8	0	0	1	0
x_9	2	0	0	1
F	0	-3	-3	-7

Ищем оптимальное решение:

Индекс разрешающего элемента: (5, 3)

	S0	x_1	x_2	x_9
x_4	1	1	1	-1
x_5	5	1	4	0
x_6	1	0	0.5	-3
x_7	0	1	0	0
x_8	0	0	1	0
x_3	2	0	0	1
F	14	-3	-3	7

Индекс разрешающего элемента: (3, 1)

	S0	x_7	x_2	x_9
x_4	1	-1	1	-1
x_5	5	-1	4	0
x_6	1	0	0.5	-3
x_1	0	1	0	0
x_8	0	0	1	0
x_3	2	0	0	1
F	14	3	-3	7

Индекс разрешающего элемента: (4, 2)

	S0	x_7	x_8	x_9
x_4	1	-1	-1	-1
x_5	5	-1	-4	0
x_6	1	0	-0.5	-3
x_1	0	1	0	0
x_2	0	0	1	0
x_3	2	0	0	1
F	14	3	3	7

x1 = 0.0

x2 = 0.0

x3 = 2.0

F = 14.0

Найдено целочисленное решение:

Ветвимся вправо по переменной x\_2 => 3

	S0	x_1	x_2	x_3
x_4	3	1	1	1
x_5	5	1	4	0
x_6	7	0	0.5	3
x_7	0	1	0	0
x_8	0	0	1	0
x_9	-3	0	0	-1
F	0	-3	-3	-7



Индекс разрешающего элемента: (3, 3)

В ветви  $x_2 \geq 3$  нет решения

Ветвимся вправо по переменной  $x_1 \Rightarrow 1$

	S0	$x_1$	$x_2$	$x_3$
$x_4$	3	1	1	1
$x_5$	5	1	4	0
$x_6$	7	0	0.5	3
$x_7$	0	1	0	0
$x_8$	-1	0	-1	0
F	0	-3	-3	-7

Индекс разрешающего элемента: (3, 2)

В ветви  $x_1 \geq 1$  нет решения

Ветвимся вправо по переменной  $x_0 \Rightarrow 1$

	S0	$x_1$	$x_2$	$x_3$
$x_4$	3	1	1	1
$x_5$	5	1	4	0
$x_6$	7	0	0.5	3
$x_7$	-1	-1	0	0
F	0	-3	-3	-7

Индекс разрешающего элемента: (3, 1)

	S0	$x_7$	$x_2$	$x_3$
$x_4$	2	1	1	1
$x_5$	4	1	4	0
$x_6$	7	0	0.5	3
$x_1$	1	-1	0	0
F	3	-3	-3	-7

Ищем оптимальное решение:

Индекс разрешающего элемента: (0, 3)

	S0	x_7	x_2	x_4
x_3	2	1	1	1
x_5	4	1	4	0
x_6	1	-3	-2.5	-3
x_1	1	-1	0	0
F	17	4	4	7

$$x_1 = 1.0$$

$$x_2 = 0$$

$$x_3 = 2.0$$

$$F = 17.0$$

Найдено целочисленное решение:

Все целочисленные решения

$$F(0.0, 0.0, 2.0) = 14.0$$

$$F(1.0, 0, 2.0) = 17.0$$

Рисунок с деревом см на картинке.

Полный перебор:

-----

$$\begin{aligned} F(0, 0, 0) &= 0 \\ F(0, 0, 1) &= 7 \\ F(0, 0, 2) &= 14 \\ F(0, 1, 0) &= 3 \\ F(0, 1, 1) &= 10 \\ F(0, 1, 2) &= 17 \\ F(1, 0, 0) &= 3 \\ F(1, 0, 1) &= 10 \\ F(1, 0, 2) &= 17 \\ F(1, 1, 0) &= 6 \\ F(1, 1, 1) &= 13 \\ F(2, 0, 0) &= 6 \\ F(2, 0, 1) &= 13 \\ F(3, 0, 0) &= 9 \end{aligned}$$

Оптимальное решение полным перебором:

$F = 17, x = (1, 0, 2)$

**Сравнение результатов от округления с найденным решением**

Результат решения ЗЛП симплекс методом:

$$\begin{aligned} x_1 &= 0.667 \\ x_2 &= 0 \\ x_3 &= 2.333 \\ F &= -18.333 \end{aligned}$$

Результат решения ЗЛП методом ветвей и границ:

$$\begin{aligned} x &= (1, 0, 2) \\ F &= 17.0 \end{aligned}$$

## Листинг

**main.py** ([https://github.com/andreev-g/iu8-decision-theory/blob/master/lab\\_3/main.py](https://github.com/andreev-g/iu8-decision-theory/blob/master/lab_3/main.py))

```
import math
import numpy as np
import pandas as pd

from src.brute import BruteForce
from src.utility import print_separator
from src.simplex.simplex import SimplexMethod

class MinTowardng:
    def __str__(self):
        return 'min'

class MaxTowardng:
    def __str__(self):
        return 'max'

class TreeNode:
    def __init__(self, f, value: SimplexMethod):
        self.left = None
        self.right = None
        self.f = f
        self.value = value

class BranchesAndBoundsMethod:
    def __init__(self, f, value):
        self.integer_solutions = []
        self.root = TreeNode(f, value)

    @staticmethod
    def find_float_idx(arr):
        if arr[0] == 'Нет решения':
            return False, 0, 0
        for idx, el in enumerate(arr[1:]):
            if not int(el) == float(el):
                return True, idx, math.floor(el)
        return False, 0, 0

    def branching(self, node):
        find, idx, el = self.find_float_idx(node.value.answer[:4])
        if find:
            # Ветвление влево если найдено дробное решение
            new_row = np.zeros(node.value.c.size)
            new_row[idx] = 1
            a = np.vstack((node.value.A, new_row))
            b = np.append(node.value.b, el)
```

```

simplex = SimplexMethod(a, b, node.value.c, mode=MaxTowardling())
try:
    print(f'Ветвимся влево по переменной  $x_{\{idx\}} \leq \{el\}$ ')
    simplex.get_result()
except AssertionError:
    node.left = TreeNode('Нет\n решения', simplex)
    print(f'В ветви  $x_{\{idx\}} \leq \{el\}$  нет решения')
    return
node.left = TreeNode(simplex.answer[0], simplex)
self.branching(node.left)
# Ветвление вправо если найдено дробное решение
new_row_right = np.zeros(node.value.c.size)
new_row_right[idx] = -1
a_right = np.vstack((node.value.A, new_row_right))
b_right = np.append(node.value.b, -(el + 1))
simplex = SimplexMethod(a_right, b_right, node.value.c, mode=MaxTowardling())
try:
    print(f'Ветвимся вправо по переменной  $x_{\{idx\}} \geq \{el+1\}$ ')
    simplex.get_result()
except AssertionError:
    node.right = TreeNode('Нет решения', simplex)
    print(f'В ветви  $x_{\{idx\}} \geq \{el+1\}$  нет решения')
    return
node.right = TreeNode(simplex.answer[0], simplex)
self.branching(node.right)
if not find:
    if node.value.answer[0] == 'Нет\n решения':
        return
    print('Найдено целочисленное решение:')
    self.integer_solutions.append(node)
    return

def start(self):
    self.branching(self.root)
    print_separator()
    print('Все целочисленные решения')
    for solution in self.integer_solutions:
        print(f"F({solution.value.answer[1]}, {solution.value.answer[2]},
{solution.value.answer[3]}) = {solution.value.answer[0]}")
    self.print()

def print(self):
    lines, *_ = self._make_string_representation(self.root)
    for line in lines:
        print(line)

def _make_string_representation(self, node):
    """
    Создает рекурсивное представление дерева
    """
    # нет ни одного поддеревя
    if node.right is None and node.left is None:
        line = '%s' % node.f

```

```

        width = len(line)
        height = 1
        middle = width // 2
        return [line], width, height, middle

# есть только левое поддерево
if node.right is None:
    lines, n, p, x = self._make_string_representation(node.left)
    s = '%s' % node.f
    u = len(s)
    first_line = (x + 1) * ' ' + (n - x - 1) * '_' + s
    second_line = x * ' ' + '/' + (n - x - 1 + u) * ' '
    shifted_lines = [line + u * ' ' for line in lines]
    return [first_line, second_line] + shifted_lines, n + u, p + 2, n + u // 2

# есть только правое поддерево
if node.left is None:
    lines, n, p, x = self._make_string_representation(node.right)
    s = '%s' % node.f
    u = len(s)
    first_line = s + x * ' ' + (n - x) * ' '
    second_line = (u + x) * ' ' + '\\' + (n - x - 1) * ' '
    shifted_lines = [u * ' ' + line for line in lines]
    return [first_line, second_line] + shifted_lines, n + u, p + 2, u // 2

# есть оба поддерева
left, n, p, x = self._make_string_representation(node.left)
right, m, q, y = self._make_string_representation(node.right)
s = '%s' % node.f
u = len(s)
first_line = (x + 1) * ' ' + (n - x - 1) * '_' + s + y * ' ' + (m - y) * ' '
second_line = x * ' ' + '/' + (n - x - 1 + u + y) * ' ' + '\\' + (m - y - 1) * ' '
if p < q:
    left += [n * ' '] * (q - p)
elif q < p:
    right += [m * ' '] * (p - q)
zipped_lines = zip(left, right)
lines = [first_line, second_line] + [a + u * ' ' + b for a, b in zipped_lines]
return lines, n + m + u, max(p, q) + 2, n + u // 2

if __name__ == '__main__':
    print('Метод ветвей и границ:')
    print_separator()

    c = [3, 3, 7]
    A = [[1, 1, 1],
          [1, 4, 0],
          [0, 0.5, 3]]
    b = [3, 5, 7]

    simplex = SimplexMethod(A, b, c, MaxToward())
    solution = simplex.get_result()

```

```
tree = BranchesAndBoundsMethod(solution[0], simplex)
tree.start()
```

```
print_separator()
print('Полный перебор:')
print_separator()
bf = BruteForce(A, b, c, solution[0])
brute_solution, value = bf.brute_optimal()
print("Оптимальное решение полным перебором:")
print(f'F = {brute_solution}, x = {value}')
```