

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
КОНФЛИКТАХ

Лабораторная работа №3 на тему:
«Формулировка и решение ЦЗЛП методом ветвей и границ»

Вариант 3

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

Москва 2021

Цель работы

Изучить постановку задачи целочисленного линейного программирования. Получить решение задачи ЦЛП методом отсекающих плоскостей.

Постановка задачи

Исходная задача ЦЛП выглядит следующим образом:

$$\begin{cases} F = cx \rightarrow \max \\ Ax \leq b \\ x_i \geq 0, \\ x_i - \text{целое}, i = \overline{1,3} \end{cases}$$

Где:

$x = (x_1 \ x_2 \ x_3)^T$ – искомый вектор решения;

$c = [8, 6, 2]$ – вектор коэффициентов целевой функции (ЦФ) F ;

$A = \begin{bmatrix} 2, & 1, & 1 \\ 1, & 4, & 0 \\ 0, & 0.5, & 1 \end{bmatrix}$,

– матрица системы ограничений;

$b = [4, 3, 6]$ – вектор правой части системы ограничений.

Требуется найти решение данной задачи ЦЛП отсекающих плоскостей.

Решение полным перебором

Метод полного перебора гарантирует нахождение оптимального решения. Однако он является крайне трудоемким в вычислительном плане, что делает его применение нежелательным.

Методом полного перебора было обнаружено оптимальное решение задачи ЦЛП:

$$F(0, 0, 0) = 0$$

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

$$F(0, 0, 2) = 4$$

$$F(0, 0, 3) = 6$$

$$F(0, 0, 4) = 8$$

$$F(1, 0, 0) = 8$$

$$F(1, 0, 1) = 10$$

$$F(1, 0, 2) = 12$$

$$F(2, 0, 0) = 16$$

Оптимальное решение полным перебором:

$$F = 16, x = (2, 0, 0)$$

Решение 3ЦЛП методом отсекающих плоскостей

Решим исходную задачу ЦЛП как задачу ЛП с помощью симплекс-метода:

Исходная симплекс-таблица:

+-----+-----+-----+-----+-----+-----+-----+-----+																
			S0		x1		x2		x3		x4		x5		x6	
-----+-----+-----+-----+-----+-----+-----+-----+																
	x4		4		2		1		1		1		0		0	
	x5		3		1		4		0		0		1		0	
	x6		6		0		0.5		1		0		0		1	
	F		0		-8		-6		-2		0		0		0	
+-----+-----+-----+-----+-----+-----+-----+-----+																

Итоговая симплекс-таблица:

		S0	x1	x2	x3	x4	x5	x6
x1	1.857	1	0	0.571	0.571	-0.143	0	
x2	0.286	0	1	-0.143	-0.143	0.286	0	
x6	5.857	0	0	1.071	0.071	-0.143	1	
F	16.571	0	0	1.714	3.714	0.571	0	

Найдено решение:

$$x_1 = 1.857$$

$$x_2 = 0.286$$

$$x_3 = 0$$

$$F = 16.571$$

Найденное решение не является целочисленным, потому добавим новое ограничение (отсекающую плоскость) в исходную задачу.

Для получения нового ограничения, необходимо найти переменную с наибольшей нецелой частью. В данном случае, этой переменной будет являться x_1 . Составим ограничение:

$$\{1\}x_1 + \{0.571\}x_3 + \{0.571\}x_4 - \{0.143\}x_5 \geq 0.857$$

$$0.571 * x_4 - 0.143 * x_5 \geq 0.857$$

И выразим это ограничение через существенные переменные:

$$x_4 = 4 - (2x_1 + x_2 + x_3)$$

$$x_5 = 3 - (x_1 + 4x_2)$$

Составим симплекс-таблицу, дополненную новым ограничением:

		S0	x1	x2	x3	x4	x5	x6	x7
x4	4	2	1	1	1	0	0	0	
x5	3	1	4	0	0	1	0	0	
x6	6	0	0.5	1	0	0	1	0	
x7	-0.857	0	0	-0.571	-0.571	-0.857	0	1	
F	0	-8	-6	-2	0	0	0	0	

Поиск опорного решения симплекс таблицы

Решим дополненную задачу симплекс-методом и получим итоговую симплекс-таблицу:

Индекс разрешающего элемента: (1, 2)

		S0	x1	x2	x3	x4	x5	x6	x7
x1	2	1	0	0.667	0.667	0	0	-0.167	
x2	0	0	1	-0.333	-0.333	0	0	0.333	
x6	6	0	0	1.167	0.167	0	1	-0.167	
x5	1	0	0	0.667	0.667	1	0	-1.167	
F	16	0	0	1.333	3.333	0	0	0.667	

Найдено решение:

$$x_1 = 2$$

$$x_2 = 0$$

$$x_3 = 0$$

$$F = 16$$

Заметим, что оно совпадает с решением найденным полным перебором. Данный факт дает основание полагать, что решение действительно является оптимальным.

Визуализация отсекающих плоскостей

Для того, чтобы ход решения задачи ЦЛП методом отсекающих плоскостей было легче понять, покажем как выглядят отсекающие плоскости, соответствующие исходным и дополнительному ограничениям.

На нижеприведенном рисунке показаны исходные ограничения в виде плоскостей в трехмерном пространстве и решение задачи ЛП в виде точки В.

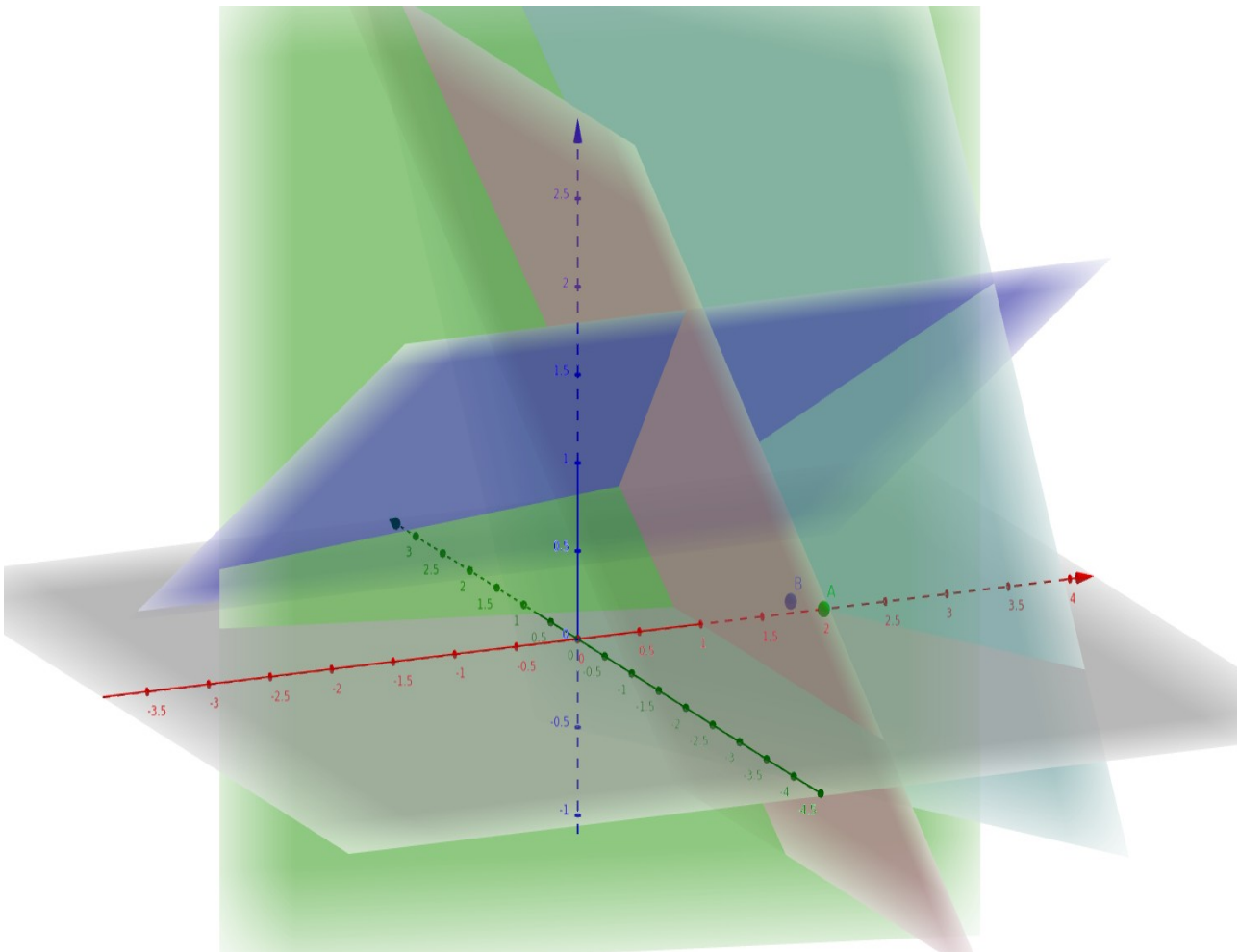


Рисунок 1. Визуализация решения задачи ЛП

Видно, что решение находится в экстремальной точке многогранника. Это соответствует действительности, ведь симплекс-метод ищет экстремальную из вершин многогранника ограничений путем обхода вершин.

Добавим найденное нами ограничение в виде плоскости красного цвета. И покажем все ограничения, точку В, соответствующую решению задачи ЛП, и точку А, соответствующую решению задачи ЦЛП.

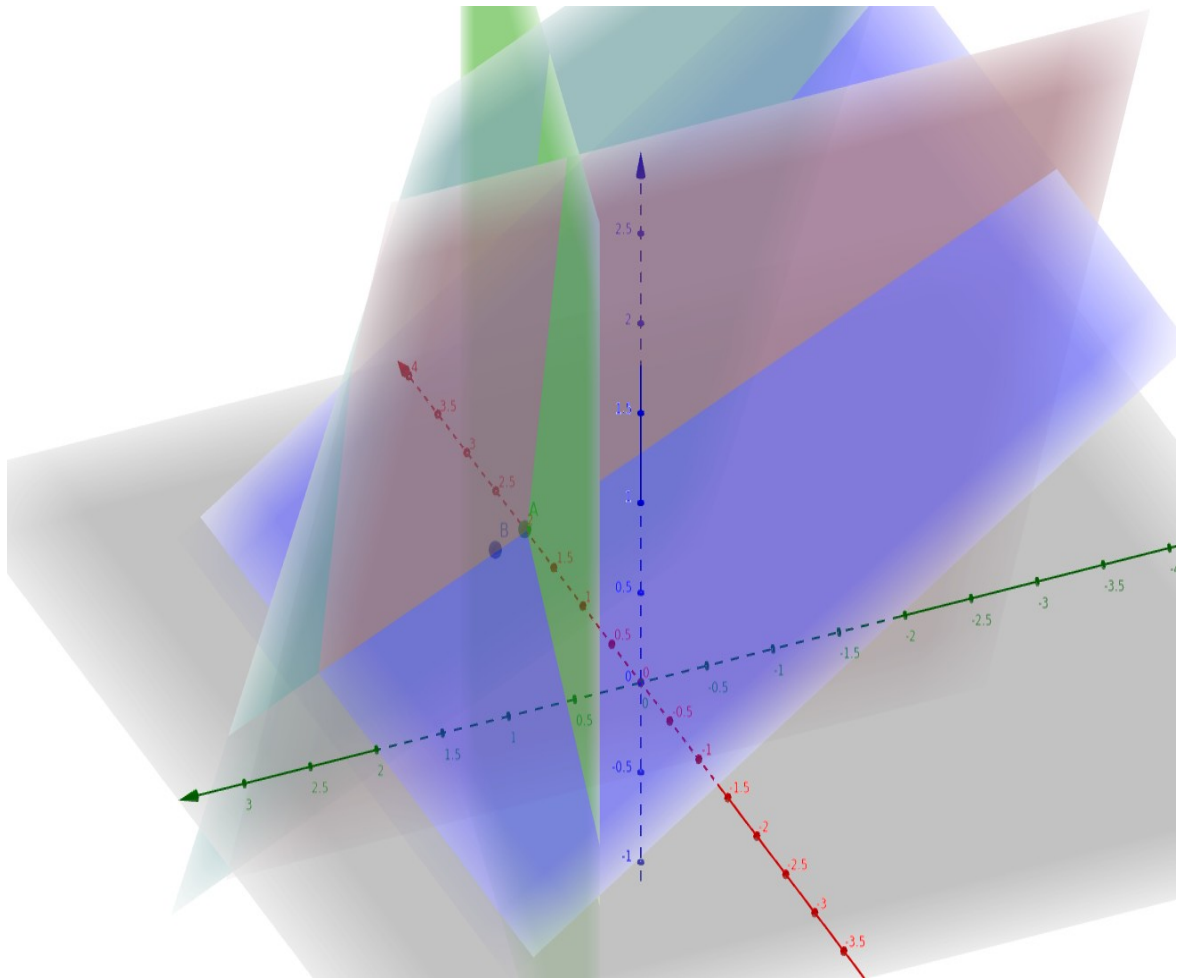


Рисунок 2. Визуализация решения задачи ЛП и ЦЛП

Целочисленное решение также попало в угол многогранника. Но в данном случае угол образован новой плоскостью, соответствующей найденному нами ограничению. Обозначения:







	$A = (2, 0, 0)$
	eq1: $2x + y + z = 4$
	eq2: $x + 4y = 3$
	eq3: $0.5y + z = 1$
	eq4: $-x - \frac{571}{1000} z = -1$
	$B = (1.86, 0.29, 0)$

Рисунок 3. Обозначения

Метод округления переменных до ближайшего целого

Решим задачу ЦЛП симплекс-методом и округлим элементы вектора решений до ближайших целых.

$$x_1 = 1.857$$

$$x_2 = 0.286$$

$$x_3 = 0$$

$$F = 16.571$$

Округлим решение:

$$x_1 = 2$$

$$x_2 = 0$$

$$x_3 = 0$$

$$F = 16$$

Методом округления было найдено целочисленное решение, которое является оптимальным. Однако теоретически метод округления не является корректным способом нахождения решения задач ЦЛП, хотя на данном конкретном примере решения совпали. Метод округления некорректен, поскольку система ограничений может вовсе не иметь решений при округлении переменных.

Вывод

В данной работе были изучены методы решения задачи ЦЛП. Конкретно были применены на практике методы полного перебора и метод отсекающих плоскостей (метод Гомори).

Листинг

main.py (https://github.com/andreev-g/iu8-decision-theory/blob/master/lab_4/main.py)

```
import numpy as np
```

```
from lab_4.simplex import Simplex
```

```
from src.utility import print_separator
```

```
class Gomory:
```

```
    def __init__(self, a, b, c):
```

```
        self.a = np.array(a)
```

```
        self.b = np.array(b)
```

```
        self.c = np.array(c)
```

```
        self.a = np.column_stack((self.a, np.eye(self.b.size)))
```

```
        self.c = np.append(self.c, np.zeros(self.b.size))
```

```
        self.simplex = Simplex(self.a, self.b, self.c, "max")
```

```
@staticmethod
```

```
def check_solution(solution):
```

```
    for el in solution:
```

```
        if not el.is_integer():
```

```
            return False
```

```
    return True
```

```
@staticmethod
```

```
def find_max_fractional_part(solution):
```

```
    solution = np.modf(solution)
```

```
    return np.argmax(solution[0])
```

```
def solution(self):
```

```
    integer_solution = False
```

```
    while not integer_solution:
```

```
        self.simplex.get_result()
```

```
        integer_solution = self.check_solution(self.simplex.answer)
```

```

if integer_solution:
    break
    idx = self.find_max_fractional_part(self.simplex.answer[:-1])
    new_limit_array = np.modf(self.simplex.matrix[idx][1:])[0]
    for index, el in enumerate(new_limit_array):
        if el < 0:
            new_limit_array[index] = el + 1
    new_limit_array *= -1
    new_limit_array[new_limit_array == -0.0] = 0.0
    new_a = np.vstack([self.simplex.A, new_limit_array])
    new_limit = np.modf(self.simplex.answer[idx])[0]
    new_b = np.append(self.simplex.b, new_limit * -1)
    added_column = np.zeros(new_b.size)
    added_column[-1] = 1
    new_a = np.column_stack((new_a, added_column))
    new_c = np.append(self.simplex.c, 0)
    self.simplex = Simplex(new_a, new_b, new_c, "max")
    print_separator()

return self.simplex.answer

```

```

if __name__ == '__main__':
    # 25
    A = [[2, 1, 1],
          [1, 4, 0],
          [0, 0.5, 1]]
    c = [8, 6, 2]
    b = [4, 3, 6]

    gomory_method = Gomory(A, b, c)
    solution = gomory_method.solution()

    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}')

```

