МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
"МИРЭА - Российский технологический университет"
РТУ МИРЭА

Институт Информационных Технологий

Кафедра Вычислительной Техники

ПРАКТИЧЕСКАЯ РАБОТА №5

по дисциплине

«Теория принятия решений»

Симплексный метод

Студент группы: ИКБО-04-22 Кликушин В.И.________
(Ф. И.О. студента)

Преподаватель __Железняк Л.М.__
(Ф.И.О. преподавателя)

Москва 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Симплексный метод – это из методов решения задач линейного программирования. Его суть заключается в сокращении перебора вершин многогранника условий. Если исследуемая вершина не отвечает условию максимума/минимума, то происходит переход к вершине, повышая/понижая значимость целевой функции. Это делается с помощью введения оценок, которые должны быть положительными, чтобы был достигнут максимум функции, и отрицательными, чтобы был достигнут минимум. С помощью этого количество перебираемых вершин существенно снижается.

1 СИМПЛЕКСНЫЙ МЕТОД

1.1 Постановка задачи

Вариант №13

Задание. Решить прямую ЗЛП с помощью симплексного метода и обратную с помощью теорем двойственности. Определить интервалы устойчивости.

Задача. В кондитерском цехе выпускают печенье двух сортов. В таблице 1 указан расход продуктов для каждого сорта и количество имеющихся продуктов.

Таблица 1. Исходные данные задачи.

Сорт	Масло	Яйца	Сахар	Молоко	Цена за 1 кг, ден. ед.
1-й сорт	0,2	0,75	0,15	0,15	1,4
2-й сорт	0,1	0,20	0,20	0,25	0,9
Запасы продуктов	100	150	100	150	

Определить, какое общее количество печенья каждого сорта надо выпекать, чтобы общая стоимость была наибольшей.

1.2 Математическая модель задачи

Пусть x1 – количество печенья первого сорта, x2 – количество печенья второго сорта. Прибыль от продажи печенья составит 1.4x1 + 0.9x2, прибыль требуется максимизировать.

Ограничения задачи:

$$\begin{cases} 0.2x1 + 0.1x2 \le 100 \\ 0.75x1 + 0.2x2 \le 150 \\ 0.15x1 + 0.2x2 \le 100 \\ 0.15x1 + 0.25x2 \le 150 \end{cases}$$

Таким образом, переходим к задаче линейного программирования:

$$f(x) = 1.4x1 + 0.9x2 \to max$$

$$\begin{cases} 0.2x1 + 0.1x2 \leq 100 \\ 0.75x1 + 0.2x2 \leq 150 \\ 0.15x1 + 0.2x2 \leq 100 \\ 0.15x1 + 0.25x2 \leq 150 \\ x_i \geq 0, i \in [1,2] \end{cases}$$

Приведем задачу к канонической форме. Для этого в левые части ограничений вводим дополнительные переменные: x3 ≥ 0. Эти переменные выбираются так, чтобы они обращали неравенства в равенства.

$$\begin{cases} 0.2x1 + 0.1x2 + x3 = 100 \\ 0.75x1 + 0.2x2 + x4 = 150 \\ 0.15x1 + 0.2x2 + x5 = 100 \\ 0.15x1 + 0.25x2 + x6 = 150 \\ x_i \geq 0, i \in [1,6] \end{cases}$$

$$f(x) = 1.4x1 + 0.9x2 + 0x3 + 0x4 + 0x5 + 0x6$$

Построим начальную симплекс-таблицу. Запишем систему в векторной форме:

$$\mathrm{A1x1} + \mathrm{A2x2} + \mathrm{A3x3} + \mathrm{A4x4} + \mathrm{A5x5} + \mathrm{A6x6} = \mathrm{A0},$$

$$\mathrm{A1} = \begin{pmatrix} 0.2 \\ 0.75 \\ 0.15 \\ 0.15 \end{pmatrix}, \mathrm{A2} = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.25 \end{pmatrix}, \mathrm{A3} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \mathrm{A4} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \ \mathrm{A5} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \mathrm{A6} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix},$$

$$\mathrm{A0} = \begin{pmatrix} 100 \\ 150 \\ 100 \\ 150 \end{pmatrix}$$

Векторы $A3$, $A4$, $A5$, A6 являются линейно независимыми единичными векторами 4-мерного пространства и образуют базис этого пространства.

Поэтому за базисные переменные выбираем переменные $x3$, $x4$, $x5$, x6. Небазисными переменными являются $x1$, $x2$. Разложение позволяет найти первое базисное допустимое решение.

Для этого свободные переменные $x1$, $x2$ приравниваем нулю. В результате получим разложение

$$A3x3 + A4x4 + A5x5 + A6x6 = A0,$$

Которому соответствует первоначальный опорный план

$$x^{(0)} = (x1, x2, x3, x4, x5, x6) == (0,0,100,150,100,150),$$
$$f\left(x^{(0)}\right) = 0.$$

Для проверки плана $x^{(0)}$ на оптимальность построим первую симплекс-таблицу. Введем в рассмотрение вектор коэффициентов целевой функции при базисных переменных.

$$\overline{C_B} = (c3, c4, c5, c6)^T = (0,0,0,0)^T.$$

В левый столбец Таблицы 1.2 запишем переменные $x3$, $x4$, x5, x6, образующие базис, в верхней строке – небазисные переменные $x1$, $x2$. В строке c_j запишем коэффициенты целевой функции, соответствующие небазисным переменным c1 = 1.4, c2 = 0.9. В столбце $\overline{C_B}$ запишем коэффициенты целевой функции, соответствующие базисным переменным Столбец, определяемый переменной $x1$, состоит из коэффициентов вектора $\overline{A_1}$. Аналогично, столбец, определяемый переменной $x2$, состоит из коэффициентов вектора $\overline{A_2}$. Крайний правый столбец заполняется элементами столбца $\overline{A_0}$, в нем же в результате вычислений получаем оптимальный план.

Заполнение f-строки (Таблица 1.3). Найдем относительные оценки Δ1, Δ2 и значение целевой функции Q.

$$\Delta_1 = (\overline{C_B} * \overline{A_1}) - c_1 = 0 * 0.2 + 0 * 0.75 + 0 * 0.15 + 0 * 0.15 - 1.4 = -1.4;$$
$$\Delta_2 = (\overline{C_B} * \overline{A_2}) - c_2 = 0 * 0.1 + 0 * 0.2 + 0 * 0.2 + 0 * 0.25 - 0.9 = -0.9;$$
$$Q = (\overline{C_B} * \overline{A_0}) = 0 * 100 + 0 * 150 + 0 * 100 + 0 * 150 = 0.$$

Таблица 1.2 – Начальная симплекс-таблица задачи о максимальном доходе

	c_j	1.4	0.9	
$\overline{C_B}$		X1	X2	$\overline{A_0}$
0	X3	0.2	0.1	100
0	X4	0.75	0.2	150
0	X5	0.15	0.2	100
0	X6	0.15	0.25	150
	f			
		Δ_1	Δ_2	Q

Таблица 1.3 – Заполнение f-строки

	c_j	1.4	0.9		
$\overline{C_B}$		X1	X2	$\overline{A_0}$	
0	X3	0.2	0.1	100	100 / 0.2 = 500
0	X4	0.75	0.2	150	150 / 0.75 = 200 min
0	X5	0.15	0.2	100	100 / 0.15 = 666.66
0	X6	0.15	0.25	150	150 / 0.15 = 1000
	f	-1.4	-0.9	0	
		Δ_1	Δ_2	Q	

Для оптимальности опорного решения в задаче на максимум требуется выполнение неотрицательности всех относительных оценок $\Delta i \geq 0$. Так как оценки $\Delta 1 = -1.4$, $\Delta 2 = -0.9$ в f-строке отрицательны, то это свидетельствуют о возможности улучшения полученного решения.

Наибольшая по модулю отрицательная оценка $\Delta 1 = -1.4$. В базис будет включена соответствующая ей небазисная переменная $x1$. Составим отношения свободных членов к положительным элементам разрешающего столбца. Данные отношения приведены справа от таблицы. Наименьшему частному соответствует строка с переменной $x4$. Эта переменная исключается из базиса. В Таблице 1.3 разрешающий столбец и разрешающая строка выделены. Разрешающим элементом является число $a21 = 0.75$.

Далее построим новую симплекс-таблицу. Ниже поэтапно демонстрируется процесс заполнения новой симплекс-таблицы (Таблицы 1.4).

Таблица 1.4 – Новая симплекс-таблица

	c_j	0	0.9	
$\overline{C_B}$		X4	X2	$\overline{A_0}$
0	X3			
1.4	X1	4/3		
0	X5			
0	X6			
	f			
		Δ_1	Δ_2	Q

В Таблице 1.4 переменные x1 и $x4$ меняются местами вместе с коэффициентами cj. Разрешающий элемент заменяется на обратный. В Таблице 1.5 элементы разрешающей строки делятся на разрешающий элемент. Элементы разрешающего столбца делятся на разрешающий элемент и меняют знак.

Таблица 1.5 – Симплекс преобразования

	c_j	0	0.9	
$\overline{C_B}$		X4	X2	$\overline{A_0}$
0	X3	-4/15		
1.4	X1	4/3	4/15	200
0	X5	-0.2		
0	X6	-0.2		
	f	28/15		
		Δ_1	Δ_2	Q

Таблица 1.6 – Итерация 0

	c_j	0	0.9		
$\overline{C_B}$		X4	X2	$\overline{A_0}$	
0	X3	-4/15	7/150	60	60 / (7/150) = 1285.7
1.4	X1	4/3	4/15	200	200 / (4/15) = 750
0	X5	-0.2	4/25	70	70 / (4/25) = 437.5 min
0	X6	-0.2	21/100	120	120 / (21/100) = 571.0 = 571.4
	f	28/15	-79/150	280	
		Δ_1	Δ_2	Q	

Остальные элементы (Таблица 1.6) рассчитываются по «правилу прямоугольника».

$$\mathrm{a}_{12}=\frac{(0.1*0.75)-(0.2*0.2)}{0.75}=\frac{7}{150};\ \mathrm{a}_{13}=\frac{(100*0.75)-(150*0.2)}{0.75}=60;$$

$$\mathrm{a}_{32}=\frac{(0.2*0.75)-(0.15*0.2)}{0.75}=\frac{4}{25};\ \mathrm{a}_{33}=\frac{(100*0.75)-(0.15*150)}{1.3}=70;$$

$$\mathrm{a}_{42}=\frac{(0.25*0.75)-(0.15*0.2)}{0.75}=\frac{21}{100};\ \mathrm{a}_{43}=\frac{(150*0.75)-(150*0.15)}{0.75}$$
$$=120;$$

$$\Delta_2=\frac{(-0.9*0.75)-(-1.4\ *\ 0.2)}{0.75}=-\frac{79}{150};$$

Базисное решение, которое дает последняя таблица

$$\mathrm{x}^{(1)}=(\mathrm{x}1,\mathrm{x}2,\mathrm{x}3,\mathrm{x}4,\mathrm{x}5,\mathrm{x}6)=(200,0{,}60{,}0{,}70{,}120),$$
$$f\left(x^{(1)}\right)=(\overline{C_B}*\overline{A_0})=0*60+1.4*200+0*70+0*120=280.$$

Это решение не является оптимальным, так как в f-строке имеется отрицательная оценка Δ2. Наибольшая по модулю отрицательная оценка Δ2 =

$-\frac{79}{150}$. В базис будет включена соответствующая ей небазисная переменная $x2$. Составим отношения свободных членов к положительным элементам разрешающего столбца. Данные отношения приведены справа от таблицы. Наименьшему частному соответствует строка с переменной $x5$. Эта переменная исключается из базиса. В Таблице 1.6 разрешающий столбец и разрешающая строка выделены. Разрешающим элементом является число $a32 = \frac{4}{25}$.

Далее построим новую симплекс-таблицу. Ниже поэтапно демонстрируется процесс заполнения новой симплекс-таблицы (Таблицы 1.7).

Таблица 1.7 – Новая симплекс-таблица

	c_j	0	0	
$\overline{C_B}$		X4	X5	$\overline{A_0}$
0	X3			
1.4	X1			
0.9	X2		25/4	
0	X6			
	f			
		Δ_1	Δ_2	Q

В Таблице 1.7 переменные $x2$ и $x5$ меняются местами вместе с коэффициентами cj. Разрешающий элемент заменяется на обратный. В Таблице 1.8 элементы разрешающей строки делятся на разрешающий элемент. Элементы разрешающего столбца делятся на разрешающий элемент и меняют знак.

Таблица 1.8 – Симплекс преобразования

	c_j	0	0	
$\overline{C_B}$		X4	X5	$\overline{A_0}$
0	X3		-7/24	
1.4	X1		-5/3	
0.9	X2	-5/4	25/4	437.5
0	X6		-21/16	
	f		79/24	
		Δ_1	Δ_2	Q

Таблица 1.9 – Итерация 1

$\overline{C_B}$	c_j	0	0	
		X4	X5	$\overline{A_0}$
0	X3	-5/24	-7/24	475/12
1.4	X1	5/3	-5/3	250/3
0.9	X2	-5/4	25/4	437.5
0	X6	1/16	-21/16	225/8
	f	29/24	79/24	6125/12
		Δ_1	Δ_2	Q

Остальные элементы (Таблица 1.9) рассчитываются по «правилу прямоугольника».

$$a_{11} = \frac{\left(-\frac{4}{15} * \frac{4}{25}\right) - \left(-0.2 * \frac{7}{150}\right)}{\frac{4}{25}} = -\frac{5}{24};$$

$$a_{13} = \frac{\left(60 * \frac{4}{25}\right) - \left(70 * \frac{7}{150}\right)}{\frac{4}{25}} = \frac{475}{12};$$

$$a_{21} = \frac{\left(\frac{4}{3} * \frac{4}{25}\right) - \left(\frac{4}{15} * -0.2\right)}{\frac{4}{25}} = \frac{5}{3};$$

$$a_{23} = \frac{\left(200 * \frac{4}{25}\right) - \left(\frac{4}{15} * 70\right)}{\frac{4}{25}} = \frac{250}{3};$$

$$a_{41} = \frac{\left(-0.2 * \frac{4}{25}\right) - \left(\frac{21}{100} * -0.2\right)}{\frac{4}{25}} = \frac{1}{16};$$

$$a_{43} = \frac{\left(120 * \frac{4}{25}\right) - \left(\frac{21}{100} * 70\right)}{\frac{4}{25}} = \frac{225}{8};$$

$$\Delta_1 = \frac{\left(\frac{28}{15} * \frac{4}{25}\right) - \left(-\frac{79}{150} * -0.2\right)}{\frac{4}{25}} = \frac{29}{24};$$

Если в последней таблице f-строке не содержит отрицательных оценок, то это свидетельствует об оптимальности полученного решения:

Базисное решение, которое дает последняя таблица

$$x^{(2)} = (x1, x2, x3, x4, x5, x6) = \left(\frac{250}{3}, 437.5, \frac{475}{12}, 0, 0, \frac{225}{8}\right),$$

$$f\left(x^{(2)}\right) = (\overline{C_B} * \overline{A_0}) = 0 * \frac{475}{12} + 1.4 * \frac{250}{3} + 0.9 * 437.5 + 0 * \frac{225}{8} = \frac{6125}{12} = 510.417.$$

Проверим решение по «правилу прямоугольника».

$$f_{max} = Q = (\overline{C_B} * \overline{A_0}) = \frac{(280 * \frac{4}{25}) - (-\frac{79}{150} * 70)}{\frac{4}{25}} = \frac{6125}{12} = 510.417.$$

Таким образом, кондитерский цех должен выпекать $x1 = \frac{250}{3}$ кг печенья первого сорта и 437.5 кг печенья второго сорта. Тогда общая стоимость будет наибольшей и кондитерская получит прибыль от продажи 510.417 [ден.ед].

1.3 Пример работы программы

```
TPR_PRACT5.csv
1   f(x) = 1.4x1 + 0.9x2
2   0.2x1 + 0.1x2 <= 100
3   0.75x1 + 0.2x2 <= 150
4   0.15x1 + 0.2x2 <= 100
5   0.15x1 + 0.25x2 <= 150
```

Рисунок 1.1 – Условия задачи в csv файле

```
Переходим к задаче линейного программирования:
f(x) = 1.4x1 + 0.9x2
{ 0.2x1 + 0.1x2 <= 100
{ 0.75x1 + 0.2x2 <= 150
{ 0.15x1 + 0.2x2 <= 100
{ 0.15x1 + 0.25x2 <= 150
```

Рисунок 1.2 – Обработка входных данных

Итерация №0

	Cj	0	0.9	
Cv		x4	x2	A0
0	x3	-0.2667	0.0467	60.0
1.4	x1	1.3333	0.2667	200.0
0	x5	-0.2	0.16	70.0
0	x6	-0.2	0.21	120.0
	f	1.8667	-0.5267	280.0

Итерация №1

	Cj	0	0	
Cv		x4	x5	A0
0	x3	-0.2083	-0.2919	39.5687
1.4	x1	1.6667	-1.6669	83.3187
0.9	x2	-1.25	6.25	437.5
0	x6	0.0625	-1.3125	28.125
	f	1.2083	3.2919	510.4313

Решение найдено! Общая прибыль составила 510.4313 денежных единиц

Рисунок 1.3 – Выполнение двух итераций и найденное решение

ЗАКЛЮЧЕНИЕ

В ходе данной работы мной был изучен симплекс-метод, произведён его ручной расчёт для решения поставленной задачи линейного программирования, а также была разработана программа на языке Python для решения задач симплекс-методом.

Плюсом метода является его универсальность, т.к. можно решать задачи линейного программирования для любого числа переменных и ограничений, однако в определённых условиях метод может уйти в полный перебор вершин области допустимых решений, что приведёт к очень долгому времени поиска решения.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Болотова Л. С. Многокритериальная оптимизация. Болотова Л. С., Сорокин А. Б. [Электронный ресурс] / Метод. указания по вып. курсовой работы — М.: МИРЭА, 2015.

2. Сорокин А. Б. Методы оптимизации: гибридные генетические алгоритмы. Сорокин А. Б. [Электронный ресурс] / Учебно-метод. пособие — М.: МИРЭА, 2016.

3. Сорокин А. Б. Линейное программирование: практикум. Сорокин А. Б., Бражникова Е. В., Платонова О. В. [Электронный ресурс] / Учебно-метод. пособие — М.: МИРЭА, 2017.

ПРИЛОЖЕНИЯ

Приложение А – Код реализации симплексного метода на языке Python.

Приложение А

Код реализации симплексного метода на языке Python.

Листинг А.1. Реализация симплексного метода.

```
import re


NUM_CRITERIA = 4  # Количество ограничений в математической моделе
PRECISION = 4  # Количество знаков после запятой при округлении
SEP = 25 # Разделитель для вывода таблицы


def print_table(system, coef_basis, coef_not_basis, basis_values,
not_basis_values):
    print(' '.ljust(SEP), end='')
    print('Cj'.ljust(SEP), end='')
    for i in range(len(coef_not_basis) + 1):
        if i == len(coef_not_basis):
            print(' '.ljust(SEP))
        else:
            print(str(coef_not_basis[i]).ljust(SEP), end='')
    print('Cv'.ljust(SEP), end='')
    for i in range(len(not_basis_values) + 1):
        if i == 0:
            print(''.ljust(SEP), end='')
        else:
            print(str(not_basis_values[i-1]).ljust(SEP), end='')
    print('A0'.ljust(SEP))
    system.insert(0, coef_basis + [' '])
    system.insert(1, basis_values + ['f'])
    for column in range(len(system[0])):
        for row in range(len(system)):
            print(str(system[row][column]).ljust(SEP), end='')
        print()
    del system[0]
    del system[0]


def get_coefficients(data):
    '''Функция для получения списка коэффициентов из системы ограничений'''
    criteria_coefficients, boundaries = list(), list()
    for exp in data:
        if '<=' in exp:
            parse = exp.split('<=')
        elif '>=' in exp:
            parse = exp.split('>=')
        elif '<' in exp:
            parse = exp.split('<')
        elif '>' in exp:
            parse = exp.split('>')
        elif '=' in exp:
            parse = exp.split('=')
        parse = list(map(str.strip, parse))
        boundaries.append(float(parse[1]))
        criteria_coefficients.append(list(map(float, [i.group(1) for i in
re.finditer(
            r'(\d+(\.\d+)?) {0,}[*]? {0,}\w', parse[0])])))
    return criteria_coefficients, boundaries
```

Продолжение Листинга А.1.

```
def count_scalar_product(vec1, vec2):
    '''Функция для рассчёта скалярного произведения двух векторов'''
    res = 0
    for i in range(len(vec1)):
        res += (vec1[i] * vec2[i])
    return res


def create_simplex_table(system, coef_basis, coef_not_basis, basis_values,
not_basis_values):
    F_str = [0] * len(not_basis_values)
    for i in range(len(not_basis_values)):
        F_str[i] = count_scalar_product(
            coef_basis, system[i]) - coef_not_basis[i]
    Q = count_scalar_product(coef_basis, system[-1])
    for i in range(len(F_str)):
        system[i].append(F_str[i])
    system[-1].append(Q)
    return F_str, Q


def simplex_iteration(system, coef_basis, coef_not_basis, basis_values,
not_basis_values, F_str, Q):
    index_column = F_str.index(min(F_str))
    mini = 1e10
    for i in range(len(system[-1]) - 1):
        tmp = system[-1][i] / system[index_column][i]
        if tmp < mini:
            mini = tmp
            index_row = i
    key_element = system[index_column][index_row]
    basis_values.insert(index_row, not_basis_values[index_column])
    not_basis_values.insert(index_column, basis_values.pop(index_row + 1))
    del not_basis_values[index_row]
    coef_basis[index_row], coef_not_basis[index_column] =
coef_not_basis[index_column], coef_basis[index_row]
    new_key_element = round(1 / key_element, PRECISION)
    data = [[0] * (len(coef_basis) + 1)
            for _ in range(len(coef_not_basis) + 1)]
    for i in range(len(system[index_column])):
        data[index_column][i] = - round(system[index_column][i] / key_element,
PRECISION)
    for i in range(len(system)):
        data[i][index_row] = round(
            system[i][index_row] / key_element, PRECISION)
    data[index_column][index_row] = new_key_element
    for row in range(len(data[0])):
        for column in range(len(data)):
            if data[column][row] == 0:
                data[column][row] = round(((system[column][row] * key_element) -
(
                    system[index_column][row] * system[column][index_row])) /
key_element, PRECISION)
    F_str = [data[i][-1] for i in range(len(data) - 1)]
    Q = data[-1][-1]
    return data, coef_basis, coef_not_basis, basis_values, not_basis_values,
F_str, Q


with open('TPR_PRACT5.csv', encoding='utf-8') as file:
    target_function = file.readline().rstrip()  # Целевая функция
    target_coefficients = list(map(float, [i.group(1) for i in re.finditer(
```

Продолжение Листинга А.1.

```
        r'(\d+(\.\d+)?) {0,}[*]? {0,}\w', target_function)]))  # Список
коэффициентов целевой функции
    criteria_function = [file.readline().rstrip() for _ in range (NUM_CRITERIA)]
    criteria_coefficients, boundaries = get_coefficients(criteria_function)
    print('Переходим к задаче линейного программирования:',
          target_function, sep='\n')
    for i in criteria_function:
        print("{ " + i)
    system = list(map(list, list(zip(*criteria_coefficients))))
    system.append(boundaries)
    # Вектор коэффициентов целевой функции при базисных переменных
    coef_basis = [0, 0, 0, 0]
    # Коэффициенты целевой функции, соответствующие небазисным переменным
    coef_not_basis = target_coefficients.copy()
    not_basis_values = re.findall(r'\w\d{1,}', target_function)
    basis_values = [f'{not_basis_values[-1][0]}{i}' for i in range(
        int(not_basis_values[-1][1]) + 1, NUM_CRITERIA + int(not_basis_values[-
1][1]) + 1)]
    F_str, Q = create_simplex_table(
        system, coef_basis, coef_not_basis, basis_values, not_basis_values)
    num_iteration = 0
    while num_iteration < 50 and min(F_str) < 0:
        print(
            ('\x1b[6;30;42m' + f"Итерация №{num_iteration}" +
'\x1b[0m').center(201))
        # print(f'Итерация №{num_iteration}'.center(201))
        system, coef_basis, coef_not_basis, basis_values, not_basis_values,
F_str, Q = simplex_iteration(
            system, coef_basis, coef_not_basis, basis_values, not_basis_values,
F_str, Q)
        print_table(system, coef_basis, coef_not_basis,
                    basis_values, not_basis_values)
        num_iteration += 1
    if num_iteration != 50:
        print(f'Решение найдено! Общая прибыль составила {Q} денежных единиц')
    else:
        print('Поставленная задача решения не имеет')
```