Изображение выглядит как эмблема, символ, герб, нашивка

Автоматически созданное описание

|  |
| --- |
| 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 3](#_Toc161609564)

[1 СИМПЛЕКСНЫЙ МЕТОД 4](#_Toc161609565)

[1.1 Постановка задачи 4](#_Toc161609566)

[1.2 Математическая модель задачи 4](#_Toc161609567)

[1.3 Пример работы программы 16](#_Toc161609568)

[ЗАКЛЮЧЕНИЕ 18](#_Toc161609569)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 19](#_Toc161609570)

[ПРИЛОЖЕНИЯ 20](#_Toc161609571)

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. **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 х1 – количество печенья первого сорта, х2 – количество печенья второго сорта. Прибыль от продажи печенья составит 1.4x1 + 0.9x2, прибыль требуется максимизировать.

Ограничения задачи:

Таким образом, переходим к задаче линейного программирования:

Приведем задачу к канонической форме. Для этого в левые части ограничений вводим дополнительные переменные: х3 ≥ 0. Эти переменные выбираются так, чтобы они обращали неравенства в равенства.

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

,

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

Которому соответствует первоначальный опорный план

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

В левый столбец Таблицы 1.2 запишем переменные 𝑥3, 𝑥4, x5, x6, образующие базис, в верхней строке – небазисные переменные 𝑥1, 𝑥2. В строке 𝑐j запишем коэффициенты целевой функции, соответствующие небазисным переменным с1 = 1.4, с2 = 0.9. В столбце запишем коэффициенты целевой функции, соответствующие базисным переменным Столбец, определяемый переменной 𝑥1, состоит из коэффициентов вектора . Аналогично, столбец, определяемый переменной 𝑥2, состоит из коэффициентов вектора . Крайний правый столбец заполняется элементами столбца , в нем же в результате вычислений получаем оптимальный план.

Заполнение f-строки (Таблица 1.3). Найдем относительные оценки ∆1, ∆2 и значение целевой функции 𝑄.

*Таблица 1.2 – Начальная симплекс-таблица задачи о максимальном доходе*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | сj | 1.4 | 0.9 |  |
|  |  | X1 | X2 |  |
| 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 |  |  |  |
|  |  | Δ1 | Δ2 | Q |

*Таблица 1.3 – Заполнение f-строки*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | сj | 1.4 | 0.9 |  |  |
|  |  | X1 | X2 |  |  |
| 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 |  |
|  |  | Δ1 | Δ2 | Q |  |

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

Наибольшая по модулю отрицательная оценка ∆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*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | сj | 0 | 0.9 |  |
|  |  | X4 | X2 |  |
| 0 | X3 |  |  |  |
| 1.4 | X1 | 4/3 |  |  |
| 0 | X5 |  |  |  |
| 0 | X6 |  |  |  |
|  | f |  |  |  |
|  |  | Δ1 | Δ2 | 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*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | сj | 0 | 0.9 |  |
|  |  | X4 | X2 |  |
| 0 | X3 | -4/15 |  |  |
| 1.4 | X1 | 4/3 | 4/15 | 200 |
| 0 | X5 | -0.2 |  |  |
| 0 | X6 | -0.2 |  |  |
|  | f | 28/15 |  |  |
|  |  | Δ1 | Δ2 | Q |

*Таблица 1.6 – Итерация 0*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | сj | 0 | 0.9 |  |  |
|  |  | X4 | X2 |  |  |
| 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 |  |
|  |  | Δ1 | Δ2 | Q |  |

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

Базисное решение, которое дает последняя таблица

Это решение не является оптимальным, так как в f-строке имеется отрицательная оценка ∆2. Наибольшая по модулю отрицательная оценка ∆2 = . 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 = .

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

*Таблица 1.7 – Новая симплекс-таблица*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | сj | 0 | 0 |  |
|  |  | X4 | X5 |  |
| 0 | X3 |  |  |  |
| 1.4 | X1 |  |  |  |
| 0.9 | X2 |  | 25/4 |  |
| 0 | X6 |  |  |  |
|  | f |  |  |  |
|  |  | Δ1 | Δ2 | 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*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | сj | 0 | 0 |  |
|  |  | X4 | X5 |  |
| 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 |  |
|  |  | Δ1 | Δ2 | Q |

*Таблица 1.9 – Итерация 1*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | сj | 0 | 0 |  |
|  |  | X4 | X5 |  |
| 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 |
|  |  | Δ1 | Δ2 | Q |

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

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

Базисное решение, которое дает последняя таблица

Проверим решение по «правилу прямоугольника».

Таким образом, кондитерский цех должен выпекать 𝑥1 = кг печенья первого сорта и кг печенья второго сорта. Тогда общая стоимость будет наибольшей и кондитерская получит прибыль от продажи 510.417 [ден.ед].

**1.3 Пример работы программы**

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

**Рисунок 1.1 – Условия задачи в csv файле**

**Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание**

**Рисунок 1.2 – Обработка входных данных**

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, Графическое программное обеспечение

Автоматически созданное описание

**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')