Министерство образования Российской Федерации

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н.Э. БАУМАНА

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

Методы оптимизации

Лабораторная работа №4 на тему:

«Решение многокритериальной оптимизации»

Вариант 5

Преподаватель:
Коннова Н.С.

Студент:
Девяткин Е.Д.

Группа:
ИУ8-34

Москва 2023

Цель работы

Изучить постановку задачи многокритериальной оптимизации (МКО); овладеть навыками решения задач МКО с помощью различных методов, выполнить сравнительный анализ результатов, полученных при помощи разных методов

Постановка задачи

Выбрать лучшую из альтернатив решения предложенной задачи по варианту из табл. с точки зрения указанных критериев следующими методами: 1) заменой критериев ограничениями; 2) формированием и сужением множества Парето; 3) методом взвешивания и объединения критериев; 4) методом анализа иерархий

Ход работы

1) Метод замени критериев ограничениями

Составим вектор весов критериев (с нашей точки зрения) используя шкалу 1÷10.

Цена	Обработка	Долговечность	Водостойкость
8	3	7	6

Нормализовав, получим $[0{,}33;\ 0{,}12;\ 0{,}29;\ 0{,}25]$;

Составим матрицу А оценок для альтернатив.

	Цена	Обработка	Долговечность	Водостойкость
Береза - A	7	8	2	2
Сосна - B	6	6	3	5
Дуб - C	1	5	6	4
Лиственница - D	3	2	7	7

Выберем в качестве главного критерия качество лечения (критерий 1).

Установим минимально допустимые уровни для остальных критериев:

Допустимое давление не менее $0{,}2 \cdot A_{max2} = 1{,}6$

Долговечность не менее $0{,}7 \cdot A_{max3} = 4{,}9$

Внешний вид не менее $0{,}5 \cdot A_{max4} = 3{,}5$

Проведём нормировку матрицы (кроме столбца главного критерия) по формуле:

$$A_{ij} = \frac{A_{ij} - A_{minj}}{A_{maxj} - A_{minj}}$$

где A_{minj} и A_{maxj} – минимальное и максимальное значение в столбце соответственно.

Нормированная матрица A:

	1	2	3	4
A	7	1	0	0
B	6	2/3	1/5	3/5
C	1	1/2	4/5	2/5
D	3	0	1	1

Проверим удовлетворение минимальным критериям. Удовлетворяют Альтернативы B, C, D. Из них выберем наиболее оптимальный вариант – A(Береза). Для данных оценок ослабление ограничений по критериям 2-4 не приведет к существенным изменениям, только если снизить требования до 0.

2) Формирование и сужения множества Парето

Выберем в качестве главных критериев для данного метода Качество лечения и уровень сервиса. Качество лечения – по оси x, уровень сервиса – по y. Сформируем множество Парето графическим методом (см. рис. 1). Оба критерия максимизируются (так как необходимо найти максимальный уровень сервиса и наилучшее качество лечения), поэтому точка утопии находится в верхнем углу графика.

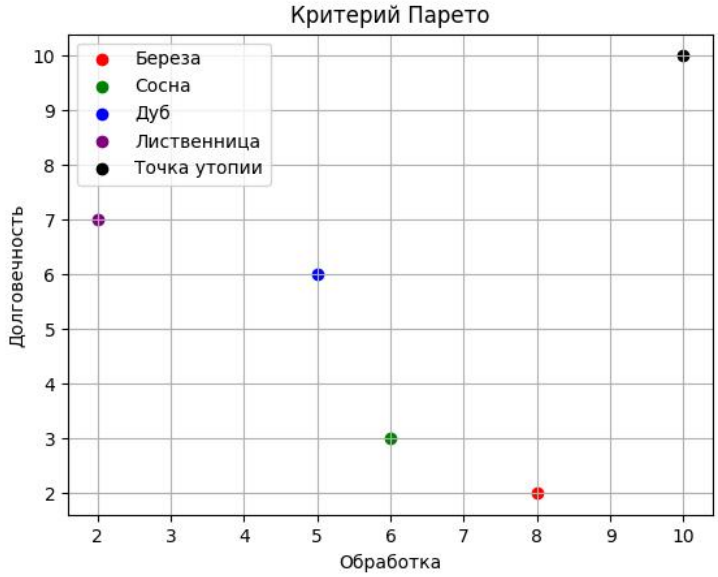


Рис. 1

Рассчитаем Манхеттенское расстояния для материалов:
1. Береза $(2 + 8 = 10)$
2. Сосна $(4 + 7 = 11)$
3. Дуб $(5 + 4 = 9)$
4. Лиственница $(8 + 3 = 11)$

Исходя из расчетов можно сказать, что Манхеттенское расстояние до точки утопии минимально для варианта С (Дуб). А значит, альтернатива С оптимальна.

3) Взвешивание и объединение критериев

Составим матрицу рейтингов альтернатив по критериям, используя шкалу 1÷10:

	1	2	3	4
A	7	8	2	2
B	6	6	3	5
C	1	5	6	4
D	3	2	7	7

Нормализуем её:

	1	2	3	4
A	0,41	0,38	0,11	0,11
B	0,35	0,28	0,16	0,28
C	0,06	0,24	0,33	0,22
D	0,18	0,10	0,39	0,39

Составим экспертную оценку критериев (по методу попарного сравнения):

$$\begin{cases} \gamma_{12} = 1; \gamma_{13} = 1; \gamma_{14} = 1 \\ \gamma_{21} = 0; \gamma_{23} = 0; \gamma_{24} = 1 \\ \gamma_{31} = 0; \gamma_{32} = 1; \gamma_{34} = 0{,}5 \\ \gamma_{41} = 0; \gamma_{42} = 0; \gamma_{43} = 0{,}5 \end{cases}$$

Получим вектор весов критериев:

$$\begin{cases} \alpha_1 = 1 + 1 + 1 = 3 \\ \alpha_2 = 0 + 0 + 0 = 0 \\ \alpha_3 = 0{,}5 + 1 + 0 = 1{,}5 \\ \alpha_4 = 0{,}5 + 1 + 0 = 1{,}5 \end{cases}$$

Нормализуем его и получим $\alpha = [0{,}50; 0; 0{,}25; 0{,}25]$

Умножим нормализованную матрицу на нормализованный вектор весов критериев и получим значения объединенного критерия для всех альтернатив:

$$\begin{pmatrix} 0{,}41 & 0{,}38 & 0{,}11 & 0{,}11 \\ 0{,}35 & 0{,}28 & 0{,}16 & 0{,}28 \\ 0{,}06 & 0{,}24 & 0{,}33 & 0{,}22 \\ 0{,}18 & 0{,}10 & 0{,}39 & 0{,}39 \end{pmatrix} \times \begin{pmatrix} 0{,}50 \\ 0 \\ 0{,}25 \\ 0{,}25 \end{pmatrix} = \begin{pmatrix} 0{,}26 \\ 0{,}28 \\ 0{,}17 \\ 0{,}28 \end{pmatrix}$$

Как видно из полученной интегральной оценки, наиболее приемлемыми являются альтернативы B и D – Сосна и Лиственница

4) <u>Метод анализа иерархий:</u>

Для каждого из критериев составим и нормализуем матрицу попарного сравнения альтернатив:

• Цена

	A	B	C	D	Сумма по строке	Нормированная сумма по строке
A	1	7/6	7	7/3	11,50	0.41
B	6/7	1	6	2	9,85	0,35
C	1/7	1/6	1	1/3	1,64	0,06
D	3/7	3/6	3	1	4,92	0,18

Отношение согласованности 1.29, сильно больше 0.1

• Обработка

	A	B	C	D	Сумма по строке	Нормированная сумма по строке
A	1	8/6	8/5	8/2	7,93	0,38
B	6/8	1	6/5	6/2	5,95	0,28
C	5/8	5/6	1	5/2	4,96	0,24
D	2/8	2/6	2/5	1	1,98	0,10

Отношение согласованности 0.41, чуть больше 0.1

• Долговечность

	A	B	C	D	Сумма по строке	Нормированная сумма по строке
A	1	2/3	2/6	2/7	2,28	0,11
B	3/2	1	3/6	3/7	3,43	0,17
C	6/2	6/3	1	6/7	6,86	0,33
D	7/2	7/3	7/6	1	8	0,39

Отношение согласованности 0.45, чуть больше 0.1

• Водостойкость

	A	B	C	D	Сумма по строке	Нормированная сумма по строке
A	1	2/5	2/4	2/7	2,18	0,11
B	5/2	1	5/4	5/7	5,46	0,28
C	4/2	4/5	1	4/7	4,37	0,22
D	7/2	7/5	7/4	1	7,65	0,39

Отношение согласованности 0.50, чуть больше 0.1

• Оценка приоритетов критериев

	A	B	C	D	Сумма по строке	Нормированная сумма по строке
A	1	8/3	8/7	8/6	6,14	0,33
B	3/8	1	3/7	3/6	2,30	0,12
C	7/8	7/3	1	7/6	5,37	0,29
D	6/8	6/3	6/7	1	4,60	0,26

Отношение согласованности -0.02

Составим матрицу (i – альтернатива, j - критерий) и умножим на столбец оценки приоритетов:

$$\begin{pmatrix} 0{,}41 & 0{,}38 & 0{,}11 & 0{,}11 \\ 0{,}35 & 0{,}28 & 0{,}16 & 0{,}28 \\ 0{,}06 & 0{,}24 & 0{,}33 & 0{,}22 \\ 0{,}18 & 0{,}10 & 0{,}39 & 0{,}39 \end{pmatrix} \times \begin{pmatrix} 0{,}33 \\ 0{,}12 \\ 0{,}29 \\ 0{,}26 \end{pmatrix} = \begin{pmatrix} 0{,}24 \\ 0{,}27 \\ 0{,}20 \\ 0{,}28 \end{pmatrix}$$

Оценив полученный вектор, можем сделать вывод, что оптимальным вариантом является D - Лиственница

Вывод

В ходе выполнения работы были изучены и разобраны различные методы решения многокритериальных задач, а именно метод замены критериев ограничениями, метод Парето, метод взвешивания и метод анализа иерархий.

Также в ходе работы в различных методах были получены различные оптимальные решения, это связано с тем, что в в первых двух методах для наглядности использовались различные главные критерии (в последних двух методах оптимальные решения оказались сравнимо одинаковы, основная проблема в погрешностях вычислений)методом Парето, взвешиванием и объединением критериев и методом анализа иерархий.

Приложение А

Ссылка на GitHub репозиторий с представленными проектами решения лабораторной работы - https://github.com/ledibonibell/MO-lab04

Приложение Б

Файл ‘Part01.py’:

```
import numpy as np


def normalize_matrix(matrix):
    normalized_matrix = np.round(matrix / np.sum(matrix), 2)

    return normalized_matrix


def find_max_min_in_columns(matrix, col_indices):
    if max(col_indices) >= matrix.shape[1]:
        raise ValueError("Некорректные индексы столбцов")

    max_min_dict = {}

    for i in col_indices:
        max_element = np.max(matrix[:, i])
        min_element = np.min(matrix[:, i])

        max_min_dict[i] = {'max': max_element, 'min': min_element}

    return max_min_dict


def multiply_max_numbers(matrix, col_indices, multiplier_array):
    max_values = [np.max(matrix[:, i]) for i in col_indices]
    result_array = np.array([max_val * multiplier_array[i] for i, max_val in
enumerate(max_values)])

    return result_array


def compute_expression(matrix, col_indices, max_min_dict, skipped_column):
    result_matrix = np.zeros_like(matrix, dtype=float)

    for i in range(matrix.shape[1]):
        if i == skipped_column:
```

```
            result_matrix[:, i] = matrix[:, i]
        elif i in col_indices:
            max_val = max_min_dict[i]['max']
            min_val = max_min_dict[i]['min']

            result_matrix[:, i] = np.round((matrix[:, i] - min_val) / (max_val - min_val), 2)

    return result_matrix


matrix_small = np.array([8, 3, 7, 6])
matrix_big = np.array([[7, 8, 2, 2],
                       [6, 6, 3, 5],
                       [1, 5, 6, 4],
                       [3, 2, 7, 7]])

print("\nМатрица приоритетов критериев:")
print(matrix_small)

normalized_matrix = normalize_matrix(matrix_small)
print("\nНормированная матрица приоритетов критериев:")
print(normalized_matrix)

print("\nМатрица материалов и критериев:")
print(matrix_big)

np.set_printoptions(formatter={'float': lambda x: "{:0.2f}".format(x)})

skipped_column_input = int(input("\nВведите номер главного критерия (от 1 до 4):
")) - 1

if skipped_column_input < 0 or skipped_column_input >= matrix_big.shape[1]:
    raise ValueError("Некорректный номер столбца")

col_indices = [i for i in range(matrix_big.shape[1]) if i != skipped_column_input]

multiplier_array = np.array([float(input(f"Введите коэффициент для неглавного
параметра №{i + 1}: ")) for i in range(3)])

result_multiply_array = multiply_max_numbers(matrix_big, col_indices,
multiplier_array)

max_min_dict = find_max_min_in_columns(matrix_big, col_indices)

print("\nМаксимальные и минимальные элементы для каждого не главного
критерия:")
for col_index, values in max_min_dict.items():
```

```
    print(f"Столбец {col_index + 1}: Максимальный элемент = {values['max']}, Минимальный элемент = {values['min']}")

print(f"\nМинимально допустимые уровни для не главных критериев:")
print(result_multiply_array)

result_matrix = compute_expression(matrix_big, col_indices, max_min_dict, skipped_column_input)
print(f"\nРезультат вычислений выражения для каждого элемента выбранных столбцов матрицы 4x4:")
print(result_matrix)
```

Файл ‘Part02.py’:

```
import numpy as np
import matplotlib.pyplot as plt

matrix_big = np.array([[7, 8, 2, 2],
                       [6, 6, 3, 5],
                       [1, 5, 6, 4],
                       [3, 2, 7, 7]])

try:
    selected_columns = [int(input('Введите номер первого главного криетрия: ')) - 1,
                        int(input('Введите номер второго главного криетрия: ')) - 1]
except ValueError:
    print("Некорректный ввод. Пожалуйста, введите целые числа")
    exit()

if any(col < 0 or col >= matrix_big.shape[1] for col in selected_columns):
    print("Некорректные номера столбцов. Пожалуйста, введите корректные номера")
    exit()

x = matrix_big[:, selected_columns[0]]
y = matrix_big[:, selected_columns[1]]

max_x_coord = x[np.argmax(matrix_big[:, selected_columns[0]])]
max_y_coord = y[np.argmax(matrix_big[:, selected_columns[1]])]

colors = ['red', 'green', 'blue', 'purple']
materials = ['Береза', 'Сосна', 'Дуб', 'Лиственница']

for i in range(len(x)):
    plt.scatter(x[i], y[i], label=materials[i], color=colors[i])
    plt.text(x[i], y[i], '', fontsize=8, ha='right', va='bottom')
```

```
plt.scatter(max_x_coord, max_y_coord, color='black', marker='o', label='Точка утопии')

plt.grid(True)

plt.xlabel(f'Критерий №{selected_columns[0]}')
plt.ylabel(f'Критерий №{selected_columns[1]}')
plt.title('Критерий Парето')
plt.legend()

plt.savefig("lab02.png")
```

Файл ‘Part03.py’:

```
import numpy as np


def normalize_matrix(matrix):
    normalized_matrix = np.round(matrix / np.sum(matrix), 2)

    return normalized_matrix


def normalize_matrix_columns(matrix):
    normalized_matrix = np.round(matrix / np.sum(matrix, axis=0), 2)

    return normalized_matrix


def pairwise_comparison(matrix):
    size = matrix.shape[0]
    result_matrix = np.zeros_like(matrix, dtype=float)

    for i in range(size):
        for j in range(i + 1, size):
            comparison_result = float(input(f"Является ли элемент [{i + 1}][{j + 1}] главным? Введите 1, 0 или 0.5: "))

            if comparison_result == 1:
                result_matrix[i, j] = 1
            elif comparison_result == 0:
                result_matrix[j, i] = 1
            elif comparison_result == 0.5:
                result_matrix[i, j] = 0.5
                result_matrix[j, i] = 0.5
```

```
            else:
                print("Некорректный ввод. Используйте 1, 0 или 0.5!")
                return None

    return result_matrix


def sum_rows(matrix):
    row_sums = np.sum(matrix, axis=1)
    return row_sums


def multiply_matrix(matrix, vector):
    result_vector = np.dot(matrix, vector)
    return result_vector


matrix_small = np.array([8, 3, 7, 6])
matrix_big = np.array([[7, 8, 2, 2],
                       [6, 6, 3, 5],
                       [1, 5, 6, 4],
                       [3, 2, 7, 7]])

print("\nМатрица приоритетов критериев:")
print(matrix_small)

normalized_matrix = normalize_matrix(matrix_small)
print("\nНормированная матрица приоритетов критериев:")
print(normalized_matrix)

print("\nМатрица материалов и критериев:")
print(matrix_big)

np.set_printoptions(formatter={'float': lambda x: "{:0.2f}".format(x)})

normalized_matrix_big = normalize_matrix_columns(matrix_big)

matrix_pairwise_comparison = pairwise_comparison(matrix_big)
result_row_sums = sum_rows(matrix_pairwise_comparison)
normalized_row_sums = normalize_matrix(result_row_sums)

result_multiply_vector = multiply_matrix(normalized_matrix_big,
normalized_row_sums)

print("\nНормированная матрица материалов и критериев:")
print(normalized_matrix_big)
```

```
if matrix_pairwise_comparison is not None:
    print("\nМатрица попарного сравнения:")
    print(matrix_pairwise_comparison)

print("\nМатрица суммы строк попарного сравнения:")
print(result_row_sums)

print("\nНормальная матрица суммы строк попарного сравнения:")
print(normalized_row_sums)

np.set_printoptions(formatter={'float': lambda x: "{:0.4f}".format(x)})

print("\nРезультат перемножения двух матриц:")
print(result_multiply_vector)
```

Файл ‘Part04.py’:

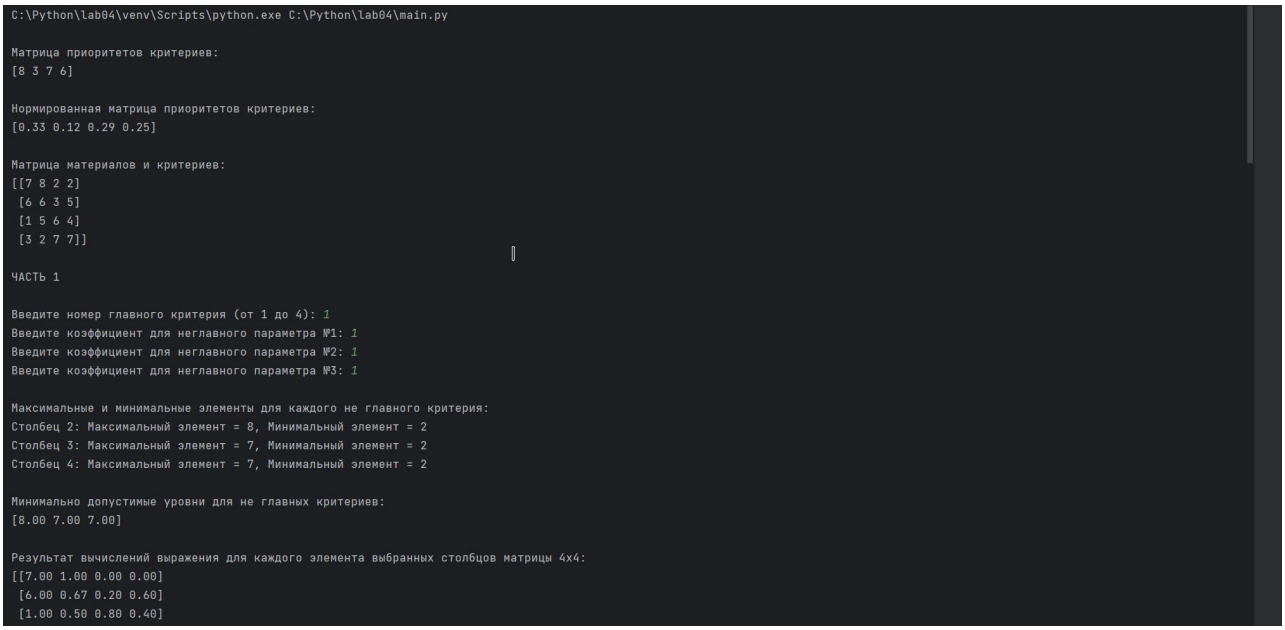
```
import numpy as np


def normalize_matrix(matrix):
    normalized_matrix = np.round(matrix / np.sum(matrix), 2)

    return normalized_matrix


def sum_rows(matrix):
    row_sums = np.sum(matrix, axis=1)
    return row_sums


def multiply_matrix(matrix, vector):
    result_vector = np.dot(matrix, vector)
    return result_vector


def normalize_matrix_another(matrix):
    size = matrix.shape[0]
    result_matrices = []

    for i in range(size):
        column_i = matrix[:, i]
        new_matrix = np.zeros_like(matrix, dtype=float)

        for j in range(size):
            new_matrix[:, j] = column_i[j] / column_i
```

```
        result_matrices.append(new_matrix.T)

    return result_matrices


matrix_small = np.array([8, 3, 7, 6])
matrix_big = np.array([[7, 8, 2, 2],
                       [6, 6, 3, 5],
                       [1, 5, 6, 4],
                       [3, 2, 7, 7]])

print("\nМатрица приоритетов критериев:")
print(matrix_small)

normalized_matrix = normalize_matrix(matrix_small)
print("\nНормированная матрица приоритетов критериев:")
print(normalized_matrix)

print("\nМатрица материалов и критериев:")
print(matrix_big)

np.set_printoptions(formatter={'float': lambda x: "{:0.2f}".format(x)})

result_new_matrix = normalize_matrix_another(matrix_big)

column1 = normalize_matrix(sum_rows(result_new_matrix[0]))
column2 = normalize_matrix(sum_rows(result_new_matrix[1]))
column3 = normalize_matrix(sum_rows(result_new_matrix[2]))
column4 = normalize_matrix(sum_rows(result_new_matrix[3]))
new_normalized_matrix = np.column_stack((column1, column2, column3, column4))

result_matrix = np.zeros((len(matrix_small), len(matrix_small)))

for i in range(len(matrix_small)):
    result_matrix[i, :] = matrix_small[i] / matrix_small

result_multiply_vector_another = multiply_matrix(new_normalized_matrix,
normalize_matrix(sum_rows(result_matrix)))

print("\nЧетыре промежуточные матрицы нормирования:")
for i, new_matrix in enumerate(result_new_matrix):
    print(f"\nМатрица {i + 1}:")
    print(new_matrix)
    print("\nСогласованность:")
    print((np.sum(normalized_matrix*(np.sum(new_matrix, axis=0))) - 4)/2.7)
    print("\nСумма строк этой матрицы")
```

```
    print(sum_rows(new_matrix))
    print("\nНормальная матрица суммы:")
    print(normalize_matrix(sum_rows(new_matrix)))

print("\nНормированная матрица другим способом:")
print(new_normalized_matrix)

print("\nПромежуточная матрица нормирования:")
print(result_matrix)

print("\nСогласованность:")
print((np.sum(normalized_matrix*(np.sum(result_matrix, axis=0))) - 4)/2.7)

print("\nНормированная матрица другим способом:")
print(normalize_matrix(sum_rows(result_matrix)))

np.set_printoptions(formatter={'float': lambda x: "{:0.4f}".format(x)})

print("\nРезультат перемножения двух матриц:")
print(result_multiply_vector_another)
```

Приложение В

Пункт 1



Пункт 2

```
 [3.00 0.00 1.00 1.00]]

ЧАСТЬ 2
Введите номер первого главного криетрия: 2
Введите номер второго главного криетрия: 3

В папке проекта лежит график lab02.png
```

Пункт 3

```
ЧАСТЬ 3
Является ли элемент [1][2] главным? Введите 1, 0 или 0.5: 1
Является ли элемент [1][3] главным? Введите 1, 0 или 0.5: 1
Является ли элемент [1][4] главным? Введите 1, 0 или 0.5: 1
Является ли элемент [2][3] главным? Введите 1, 0 или 0.5: 0
Является ли элемент [2][4] главным? Введите 1, 0 или 0.5: 0
Является ли элемент [3][4] главным? Введите 1, 0 или 0.5: 0.5

Нормированная матрица материалов и критериев:
[[0.41 0.38 0.11 0.11]
 [0.35 0.29 0.17 0.28]
 [0.06 0.24 0.33 0.22]
 [0.18 0.10 0.39 0.39]]

Матрица попарного сравнения:
[[0.00 1.00 1.00 1.00]
 [0.00 0.00 0.00 0.00]
 [0.00 1.00 0.00 0.50]
 [0.00 1.00 0.50 0.00]]

Матрица суммы строк попарного сравнения:
[3.00 0.00 1.50 1.50]

Нормальная матрица суммы строк попарного сравнения:
[0.50 0.00 0.25 0.25]

Результат перемножения двух матриц:
[0.2600 0.2875 0.1675 0.2850]
```

Пункт 4

```
ЧАСТЬ 4

Четыре промежуточные матрицы нормирования:

Матрица 1:
[[1.00 1.17 7.00 2.33]
 [0.86 1.00 6.00 2.00]
 [0.14 0.17 1.00 0.33]
 [0.43 0.50 3.00 1.00]]

Согласованность:
1.2918871252204585

Сумма строк этой матрицы
[11.50 9.86 1.64 4.93]

Нормальная матрица суммы:
[0.41 0.35 0.06 0.18]

Матрица 2:
[[1.00 1.33 1.60 4.00]
 [0.75 1.00 1.20 3.00]
 [0.62 0.83 1.00 2.50]
 [0.25 0.33 0.40 1.00]]

Согласованность:
0.4182407407407407

Сумма строк этой матрицы
[7.93 5.95 4.96 1.98]

Нормальная матрица суммы:
Матрица 3:
[[1.00 0.67 0.33 0.29]
 [1.50 1.00 0.50 0.43]
 [3.00 2.00 1.00 0.86]
 [3.50 2.33 1.17 1.00]]

Согласованность:
0.44550264550264546

Сумма строк этой матрицы
[2.29 3.43 6.86 8.00]

Нормальная матрица суммы:
[0.11 0.17 0.33 0.39]

Матрица 4:
[[1.00 0.40 0.50 0.29]
 [2.50 1.00 1.25 0.71]
 [2.00 0.80 1.00 0.57]
 [3.50 1.40 1.75 1.00]]

Согласованность:
0.49994708994709

Сумма строк этой матрицы
[2.19 5.46 4.37 7.65]

Нормальная матрица суммы:
[0.11 0.28 0.22 0.39]
Нормированная матрица другим способом:
[[0.41 0.38 0.11 0.11]
 [0.35 0.29 0.17 0.28]
 [0.06 0.24 0.33 0.22]
 [0.18 0.10 0.39 0.39]]

Промежуточная матрица нормирования:
[[1.00 2.67 1.14 1.33]
 [0.38 1.00 0.43 0.50]
 [0.88 2.33 1.00 1.17]
 [0.75 2.00 0.86 1.00]]

Согласованность:
-0.02063492063492075

Нормированная матрица другим способом:
[0.33 0.12 0.29 0.25]

Результат перемножения двух матриц:
[0.2403 0.2696 0.1993 0.2828]
```