Министерство образования Российской Федерации

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Н.Э. БАУМАНА

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

ТЕОРИЯ СИСТЕМ И СИСТЕМНЫЙ АНАЛИЗ

Лабораторная работа №6 на тему:
«Решение задачи многокритериальной оптимизации»

Вариант 7

Преподаватель:
Строганов И.С.

Студент:

Заботин Д.В.

Группа:
ИУ8-31

Москва 2021

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

Исходные данные

Выбор спутника жизни:

A. Анатолий;

B. Александр;

C. Владимир;

D. Сергей;

Критерии:

1 Образование;

2 Физическая подготовка;

3 Внешность;

4 Характер;

Описание предпочтений:

Образование: Сергей учится в аспирантуре, Александр закончил технический вуз, Владимир — военное училище, Анатолий — экономический колледж.

Физподготовка: самый физически крепкий — Владимир, Александр и Сергей уступают немного, Анатолий —существенно.

Внешность: Анатолий и Владимир — красавцы, Александр и Сергей — довольно симпатичны.

Характер: Анатолий — повеса и сердцеед, Александр — оптимист и трудяга, Владимир — лидер и карьерист, Сергей — умница и самоед

Постановка задачи

Выбрать лучшую из альтернатив решения предложенной задачи 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

Составим вектор весов критериев, используя шкалу 1÷10

Начальная цена	Стоимость обслуживания	Объем памяти	Размер экрана
8	6	2	4

Таблица 1 - Веса критериев.

Нормализовав, получим вектор (0.4, 0.3, 0.1, 0.2).

1) Метод замены критериев ограничениями

Составим матрицу A оценок для альтернатив.

Критерии / Альтернатива	1	2	3	4
A	3	9	1	8
B	5	6	3	9
C	7	4	4	4
D	9	1	1	1

Таблица 2 - Матрица оценок для альтернатив.

В качестве главного критерия выберем физическую подготовку (критерий 2).

Установим минимально допустимые уровни для остальных критериев:

1 Образование — не менее $0.4 * A_{max_1}$

2 Внешность — не менее $0.6 * A_{max_3}$

3 Характер — не менее $0.6 * A_{max_4}$

Проведем нормировку матрицы.

Критерии / Альтернатива	1	2	3	4
A	0	**9**	0	0.875
B	0.333	**6**	0.666	1
C	0.666	**4**	1	0.375
D	1	**1**	0	0

Таблица 3 - нормированная матрица оценок для альтернатив

При заданных условиях лучшее решение отсутствует.

2) Формирования и сужение множества Парето

Выберем в качестве главных критериев для данного метода образование (критерий №1) и внешность (критерий №3). Образование – ось Ox, внешность – ось Oy. Сформируем множество Парето графическим методом. Оба критерия максимизируются, поэтому точка утопии находится в правом верхнем углу.

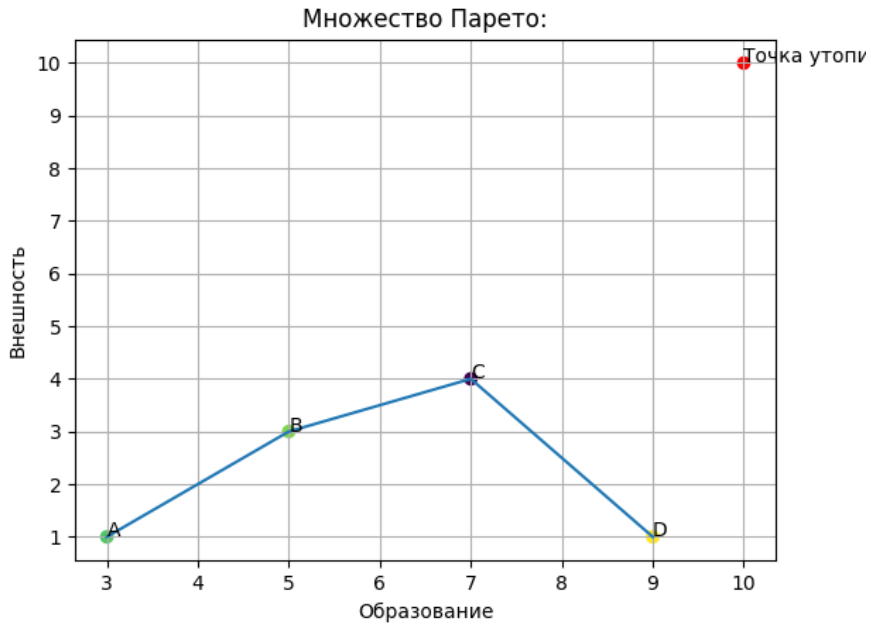

Рисунок 1 - Графическое решение методом сужения множества Парето.

По графику можно увидеть, что Евклидово расстояние до точки утопии минимально для варианта C – Владимир. Значит эта альтернатива оптимальна.

3) Взвешивание и объединение критериев

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

Критерии / Альтернатива	1	2	3	4
A	3	9	1	8
B	5	6	3	9
C	7	4	4	4
D	9	1	1	1

Таблица 4 - Матрица оценок для альтернатив.

Нормализуем её.

Критерии / Альтернатива	1	2	3	4
A	0.125	0.450	0.111	0.364
B	0.208	0.300	0.333	0.409
C	0.292	0.200	0.444	0.182
D	0.375	0.050	0.111	0.045

Таблица 5 - Нормализованная матрица оценок для альтернатив.

Составим экспертную оценку критериев (по методу парного сравнения):

$\gamma_{12} = 1$, $\gamma_{13} = 0$, $\gamma_{14} = 0$, $\gamma_{23} = 0.5$, $\gamma_{24} = 0.5$, $\gamma_{34} = 0.5$

Получим вектор весов критериев:

$a_1 = 1+0+0 = 1$

$a_2 = 1+0.5+0.5 = 2$

$a_3 = 0+0.5+0.5 = 1$

$a_4 = 0+0.5+0.5 = 1$

Нормализуем вектор и получим α = (0.2, 0.25, 0.25, 0.3)

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

$$\begin{pmatrix} 0.125 & 0.450 & 0.111 & 0.364 \\ 0.208 & 0.300 & 0.333 & 0.409 \\ 0.292 & 0.200 & 0.444 & 0.182 \\ 0.200 & 0.050 & 0.111 & 0.045 \end{pmatrix} * \begin{pmatrix} 0.2 \\ 0.25 \\ 0.25 \\ 0.3 \end{pmatrix} = \begin{pmatrix} 0.274 \\ 0.323 \\ 0.274 \\ 0.129 \end{pmatrix}$$

Как следует из полученной оценки, наиболее приемлемой является альтернатива B – Александр.

4) <u>Метод анализа иерархий</u>

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

Матрица для критерия "Образование":

A	B	C	D	GM	NPV
1.000	1.000	0.200	0.143	0.411	0.081
3.000	1.000	0.333	0.143	0.615	0.121
7.000	3.000	1.000	0.333	1.627	0.320
7.000	5.000	1.000	1.000	2.432	0.478

Отношение согласованности: 0.092 < 0.1

Скриншот 1 - Образование.

Матрица для критерия "Физическая подготовка":

A	B	C	D	GM	NPV
1.000	3.000	7.000	7.000	3.482	0.575
0.333	1.000	5.000	5.000	1.699	0.281
0.200	0.200	1.000	3.000	0.589	0.097
0.143	0.143	0.333	1.000	0.287	0.047

Отношение согласованности: 0.087 < 0.1

Скриншот 2 - Физическая подготовка.

```
Матрица для критерия "Внешность":
A     | B     | C     | D     | GM    | NPV
1.000 | 5.000 | 1.000 | 5.000 | 2.236 | 0.405
0.200 | 1.000 | 0.143 | 1.000 | 0.411 | 0.074
1.000 | 7.000 | 1.000 | 5.000 | 2.432 | 0.440
0.200 | 1.000 | 0.200 | 1.000 | 0.447 | 0.081
Отношение согласованности: 0.005 < 0.1
```

Скриншот 3 - Внешность.

```
Матрица для критерия "Характер":
A     | B     | C     | D     | GM    | NPV
1.000 | 1.000 | 5.000 | 5.000 | 2.236 | 0.417
1.000 | 1.000 | 5.000 | 5.000 | 2.236 | 0.417
0.200 | 0.200 | 1.000 | 1.000 | 0.447 | 0.083
0.200 | 0.200 | 1.000 | 1.000 | 0.447 | 0.083
Отношение согласованности: 0.0 < 0.1
```

Скриншот 4 - Характер.

```
Оценка приоритетов критериев:
A     | B     | C     | D     | GM    | NPV
1.000 | 5.000 | 3.000 | 7.000 | 3.201 | 0.578
0.333 | 1.000 | 0.333 | 5.000 | 0.863 | 0.156
0.200 | 3.000 | 1.000 | 3.000 | 1.158 | 0.209
0.143 | 0.333 | 0.200 | 1.000 | 0.312 | 0.056
Отношение согласованности: 0.103 незначительно больше 0.1
```

Скриншот 5 - Оценка приоритетов критериев.

Алгоритм применения метода анализа иерархий:

1) Рассчитываем значение среднего геометрического в каждой строке

текущей матрицы по формуле:
$$\begin{gathered} gm_1 = \sqrt[n]{a_{1_1} * a_{1_2} * ... * a_{1_m}} \\ gm_2 = \sqrt[n]{a_{2_1} * a_{2_2} * ... * a_{2_m}}, \\ ..... \\ gm_n = \sqrt[n]{a_{n_1} * a_{n_2} * ... * a_{n_m}} \end{gathered}$$

где n — количество строк (количество критериев);

m — количество столбцов;

a — элемент матрицы.

2) Далее вычисляем сумму средних геометрических:

$$\sum_{i=1}^{n} gm_i = gm_1 + gm_1 + ... + gm_n$$

3) Зная среднее геометрического строки матрицы и сумму средних геометрических, мы можем вычислить значения компонент нормализованного вектора приоритетности (НВП):

1-ая компонента НВП для текущей матрицы: $\frac{gm_1}{\sum_{i=1}^{n} gm_i}$

2-ая компонента НВП для текущей матрицы: $\frac{gm_2}{\sum_{i=1}^{n} gm_i}$

.....

n-ая компонента НВП для текущей матрицы: $\frac{gm_n}{\sum_{i=1}^{n} gm_i}$

4) Получив все компоненты НВП, мы можем найти собственное значение матрицы:

$$\begin{gathered} \lambda_{\max} = (\sum \textit{элементы } 1-\textit{го столбца матрицы}) \times (1-\textit{ая компонента НВП}) + \\ + (\sum \textit{элементы } 2-\textit{го столбца матрицы}) \times (2-\textit{ая компонента НВП}) + \\ ..... \\ + (\sum \textit{элементы } n-\textit{го столбца матрицы}) \times (n-\textit{ая компонента НВП}) \end{gathered}$$

5) Теперь мы можем рассчитать индекс согласования:

$$\text{ИС} = \frac{\lambda_{max} - n}{n-1},$$

где n — количество столбцов матрицы (количество критериев).

6) Зная ИС и ПСС (показатель случайной согласованности) = 0.9 (для 4-х критериев, размер матрицы 4x4), мы можем посчитать отношение согласованности ОС:

$$OC = \frac{ИС}{ПСС}$$

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

$$\begin{pmatrix} 0.081 & 0.575 & 0.405 & 0.417 \\ 0.121 & 0.281 & 0.074 & 0.074 \\ 0.320 & 0.097 & 0.440 & 0.083 \\ 0.478 & 0.047 & 0.081 & 0.083 \end{pmatrix} * \begin{pmatrix} 0.578 \\ 0.156 \\ 0.209 \\ 0.056 \end{pmatrix} = \begin{pmatrix} 0.245 \\ 0.153 \\ 0.297 \\ 0.306 \end{pmatrix}$$

Оценив полученный вектор, можем сделать вывод, что оптимальным вариантом является вариант D – Сергей.

Выводы

В результате проделанной работе были изучены различные методы решения задач многокритериальной оптимизации: «Метод главного критерия», «Метод формирования и сужения множества Парето», «Метод объединения критериев», «Метод анализа иерархий». Результат вычислений в задании 1 показал, что при выбранных ограничениях оптимального решения нет, но если, например, выбрать следующий вектор ограничений: [0.6, 0, 0.4, 0,3], то результатом 1-го метода будет C — Владимир. Для метода с множеством Парето оптимальным решением является альтернатива C — Владимир. Эти результаты отличается от задания 3, где лучшим результатом является вариант B — Александр. В задании 4 ответ — альтернатива D — Сергей.

Программная реализация методов представлена в **приложении 1**.

Приложение 1:

Файл major_criterion.py:

```
# Copyright 2021 DimaZzZz101 zabotin.d@list.ru

import functions as fn
import numpy as np

"""
                                Лабораторная работа №6
                            Решение задач многокритериальной оптимизации
        Цель работы: изучить постановку задачи многокритериальной оптимизации (МКО); овладеть навыками решения
                задач МКО с помощью различных методов, выполнить сравнительный анализ результатов,
                полученных с помощью разных методов.
                                Вариант 7
"""

# Задание 1.
if __name__ == '__main__':
    print('Метод главного критерия:', end='\n\n')

    alternatives = {
        'A': "Анатолий",
        'B': "Александр",
        'C': "Владимир",
        'D': "Сергей",
    }
```

```
    names = np.array(list('ABCD'))

    # Вектор весов критериев.
    criteria_weights = np.array([8, 6, 2, 4])

    # Нормальзуем вектор весов критериев.
    normalized_criteria_weights = np.divide(criteria_weights,
np.sum(criteria_weights))

    # Матрица оценок альтернатив.
    rating_matrix = np.array([[3., 9., 1., 8.],
                              [5., 6., 3., 9.],
                              [7., 4., 4., 4.],
                              [9., 1., 1., 1.]])

    normed_matrix = fn.ForMajorCriterion(rating_matrix,
1).matrix_normalization()

    # Минимально допутсимые уровни для критериев.
    lower_thresholds = np.array([0.4, 1, 0.6, 0.6])

    # Заменяем критерии на ограничения для приска подходящего решения.
    indices = fn.criteria_to_restrictions(normed_matrix, lower_thresholds)

    # Вывод результатов.
    print('Вектор весов критериев:', criteria_weights, end='\n\n')
    print('Нормализованный вектор весов критериев: ',
np.round(normalized_criteria_weights, 2), end='\n\n')
    print('Ограничения на веса: ', lower_thresholds, end='\n\n')
```

```
    print('Матрица оценок альтернатив:')
    for i, row in zip(names, rating_matrix):
        print(str(i), *row, sep=' | ')
    print()

    print('Нормированная матрица:')
    for i, row in zip(names, normed_matrix):
        print(i, *[format(e, '.3f') for e in row], sep=' | ')
    print()

    print('Выбранная альтернатива:')
    for i, row in zip(names[indices], rating_matrix[indices]):
        print(f'{alternatives[i]}:')
        print('\t', str(i), *row, sep=' | ')
```

Файл pareto_set.py:

```
# Copyright 2021 DimaZzZz101 zabotin.d@list.ru

import functions as fn
import numpy as np

"""
                                Лабораторная работа №6
                            Решение задач многокритериальной оптимизации
        Цель работы: изучить постановку задачи многокритериальной оптимизации (МКО); овладеть навыками решения
                задач МКО с помощью различных методов, выполнить сравнительный анализ результатов,
                полученных с помощью разных методов.
```

Вариант 7

"""

```
# Задание 2.
if __name__ == '__main__':
    print('Формирование множества Парето:', end='\n\n')

    alternatives = {
        'A': "Анатолий",
        'B': "Александр",
        'C': "Владимир",
        'D': "Сергей",
    }

    names = np.array(list('ABCD'))

    # Матрица оценок альтернатив.
    rating_matrix = np.array([[3, 9, 1, 8],
                              [5, 6, 3, 9],
                              [7, 4, 4, 4],
                              [9, 1, 1, 1]])

    # Получаем массив расстояний до от каждой точки с координатами из двух критериев до точки утопии.
    dist = fn.create_pareto_set(rating_matrix, [0, 2], (10, 10), ['Образование', 'Внешность'], names)

    # Получаем индекс решения с минимальным расстоянием до точки утопии.
    index = np.argmin(dist)
```

```
    # Вывод результатов.
    print('Матрица критериев:')
    for i, row in zip(names, rating_matrix):
        print(i, *row, sep=' | ')
    print()
    print('Евклидово расстояние:')
    for i, d in zip(names, dist):
        print(i, d, sep=' | ')
    print()
    print('Альтернатива: ', alternatives[names[index]], names[index],
*rating_matrix[index], sep=' | ')
```

Файл union.py:

```
# Copyright 2021 DimaZzZz101 zabotin.d@list.ru

import functions as fn
import numpy as np

"""
                                Лабораторная работа №6
                            Решение задач многокритериальной оптимизации
        Цель работы: изучить постановку задачи многокритериальной
оптимизации (МКО); овладеть навыками решения
                задач МКО с помощью различных методов, выполнить
сравнительный анализ результатов,
                полученных с помощью разных методов.
                                Вариант 7
"""
```

```
# Задание 3.
if __name__ == '__main__':
    print('Взвешивание и объединение критериев:', end='\n\n')

    alternatives = {
        'A': "Анатолий",
        'B': "Александр",
        'C': "Владимир",
        'D': "Сергей",
    }

    names = np.array(list('ABCD'))

    # Матрица оценок альтернатив.
    criteria_matrix = np.array([[3, 9, 1, 8],
                                [5, 6, 3, 9],
                                [7, 4, 4, 4],
                                [9, 1, 1, 1]])

    comparisons = [1, 0, 0, 0.5, 0.5, 0.5]

    # Нормализация матрицы.
    sums = np.sum(np.transpose(criteria_matrix), axis=1)
    criteria_matrix = np.divide(criteria_matrix, sums)

    # Создание массива весов из попарных сравнений.
    weights = fn.create_weights_matrix(4, comparisons, lambda x: 1 - x)

    # Вектор весов критериев.
    weights = np.sum(weights, axis=1)
```

```
    # Нормализация ветора весов критериев.
    weights = weights / np.sum(weights)

    # Умножение нормализованной матрицы на нормализованный вектор весов критериев.
    # В результате получим значение объединенного критерия для всех альтернатив.
    result = np.dot(criteria_matrix, weights)

    # Вывод результатов.
    print('Нормализированный вектор весов: ', weights, end='\n\n')

    print('Нормализированная матрица критериев:')
    for i, row in zip(names, criteria_matrix):
        print(i, *[format(e, '.3f') for e in row], sep=' | ')
    print()
    print('Объединенные критерии:')
    for i, r in zip(names, result):
        print(i, format(r, '.3f'), sep=' | ')
    print()
    print('Альтернатива: \n\t', names[result.argmax()], ' - ', alternatives[names[result.argmax()]])
```

Файл union.py:

```
# Copyright 2021 DimaZzZz101 zabotin.d@list.ru

import functions as fn
```

```
import numpy as np

"""
                                Лабораторная работа №6
                                Решение задач многокритериальной оптимизации
        Цель работы: изучить постановку задачи многокритериальной оптимизации (МКО); овладеть навыками решения
                задач МКО с помощью различных методов, выполнить сравнительный анализ результатов,
                полученных с помощью разных методов.
                                Вариант 7
"""


def print_matrix(matrix, columns=None, form='.3f', sep=' | '):
    if columns is not None:
        print(*columns, sep=sep)
    for row in matrix:
        print(*[format(e, form) for e in row], sep=sep)


# Задание 4.
if __name__ == '__main__':
    print('Метод анализа иерархий:', end='\n\n')

    alternatives = {
        'A': "Анатолий",
        'B': "Александр",
        'C': "Владимир",
        'D': "Сергей",
```

```
    }

    # Показатель случайной согласованности.
    # Random consistency score.
    rcs = 0.90

    names = np.array(list('ABCD'))
    # GM - geometry mean - среднее геометрическое.
    # NPV - normalized priority vector - нормализованный вектор приоритетов.
    columns = ['A    ', 'B    ', 'C    ', 'D    ', 'GM   ', 'NPV  ']
    print('Матрица для критерия "Образование": ')
    criteria = [1, 1 / 5, 1 / 7, 1 / 3, 1 / 7, 1 / 3]
    criteria_weights = fn.create_weights_matrix(4, criteria, lambda x: 1 / x)
    means, npv_1 = fn.normalized_priority_vector(criteria_weights)
    print_matrix(np.hstack([criteria_weights, means[:, np.newaxis], npv_1[:,
np.newaxis]]), columns)
    accordance = fn.count_accordance(criteria_weights, npv_1, rcs)
    print('Отношение согласованности: {}'.format(round(accordance, 3)), '< 0.1',
end='\n\n')

    print('Матрица для критерия "Физическая подготовка": ')
    criteria = [3, 7, 7, 5, 5, 3]
    criteria_weights = fn.create_weights_matrix(4, criteria, lambda x: 1 / x)
    means, npv_2 = fn.normalized_priority_vector(criteria_weights)
    print_matrix(np.hstack([criteria_weights, means[:, np.newaxis], npv_2[:,
np.newaxis]]), columns)
    accordance = fn.count_accordance(criteria_weights, npv_2, rcs)
    print('Отношение согласованности: {}'.format(round(accordance, 3)), '< 0.1',
end='\n\n')
```

```
    print('Матрица для критерия "Внешность": ')
    criteria = [5, 1, 5, 1 / 7, 1, 5]
    criteria_weights = fn.create_weights_matrix(4, criteria, lambda x: 1 / x)
    means, npv_3 = fn.normalized_priority_vector(criteria_weights)
    print_matrix(np.hstack([criteria_weights, means[:, np.newaxis], npv_3[:,
np.newaxis]]), columns)
    accordance = fn.count_accordance(criteria_weights, npv_3, rcs)
    print('Отношение согласованности: {}'.format(round(accordance, 3)), '< 0.1',
end='\n\n')

    print('Матрица для критерия "Характер": ')
    criteria = [1, 5, 5, 5, 5, 1]
    criteria_weights = fn.create_weights_matrix(4, criteria, lambda x: 1 / x)
    means, npv_4 = fn.normalized_priority_vector(criteria_weights)
    print_matrix(np.hstack([criteria_weights, means[:, np.newaxis], npv_4[:,
np.newaxis]]), columns)
    accordance = fn.count_accordance(criteria_weights, npv_4, rcs)
    print('Отношение согласованности: {}'.format(round(accordance, 3)), '< 0.1',
end='\n\n')

    print('Оценка приоритетов критериев: ')
    criteria = [5, 3, 7, 1 / 3, 5, 3]
    criteria_weights = fn.create_weights_matrix(4, criteria, lambda x: 1 / x)
    means, npv = fn.normalized_priority_vector(criteria_weights)
    print_matrix(np.hstack([criteria_weights, means[:, np.newaxis], npv[:,
np.newaxis]]), columns)
    accordance = fn.count_accordance(criteria_weights, npv, rcs)
    print('Отношение согласованности: {}'.format(round(accordance, 3)),
'незначительно больше 0.1', end='\n\n')
```

```
    print('Нормальная матрица приоритетов:')
    npv_matrix = np.transpose([npv_1, npv_2, npv_3, npv_4])
    print_matrix(npv_matrix)
    print()
    print('Результирующий вектор')
    for i, j in zip(names, np.dot(npv_matrix, npv)):
        print(i, format(j, '.3f'), sep='|')
    print()
    print('Альтернатива:')
    alt = names[np.dot(npv_matrix, npv).argmax()]
    print(f'\t{alternatives[alt]}: {alt}')
```

Файл functions.py:

```
# Copyright 2021 DimaZzZz101 zabotin.d@list.ru

import matplotlib.pyplot as plt
import numpy as np
import math as m

"""
                    Лабораторная работа №6
                Решение задач многокритериальной оптимизации
    Цель работы: изучить постановку задачи многокритериальной
оптимизации (МКО); овладеть навыками решения
            задач МКО с помощью различных методов, выполнить
сравнительный анализ результатов,
            полученных с помощью разных методов.
                    Вариант 7
"""
```

```
# По варианту:
# Задача: Выбор спутника жизни;
# Поиск расстояния: Евклидова метрика;

# Альтенативы:
# A. Анатолий;
# B. Александр;
# C. Владимир;
# D. Сергей;

# Критерии:
# 1. Образование;
# 2. Физическая подготовка;
# 3. Внешность;
# 4. Характер;

# етод замены критериев ограничениями
class ForMajorCriterion:
    def __init__(self, matrix, index):
        self.matrix = matrix.copy()
        self.index = index

    def matrix_normalization(self):
        min_ = np.min(self.matrix, axis=0)
        max_ = np.max(self.matrix, axis=0)
        for i in range(4):
            if i != self.index:
                for j in range(4):
```

```
                self.matrix[j][i] = (self.matrix[j][i] - min_[i]) / (max_[i] - min_[i])
        return self.matrix


# Возврат индексов жизнеспособных альтернатив по порядку на основе главного критерия.
def criteria_to_restrictions(weights, restrictions):
    assert 1. in restrictions, "Отсутствует главный критерий."
    # Запоминаем столбец с главным критерием.
    main = np.where(restrictions == 1.)
    assert len(main), "Главный критерий может быть только один."
    # Транспонируем матрицу и находим максимальные элементы в строках этой матрицы.
    maxes = np.max(np.transpose(weights), axis=1)
    # Индикаторы перехода выше допустимого ограничения.
    indicators = np.array([weight >= restrictions * maxes for weight in weights.astype(float)])
    # Чтобы ориентироваться по главному критерю переводим все критерии до него в true.
    indicators[:, main] = True
    # Построчно берем ндексы тех строк, которые равны true.
    idx_list = np.flatnonzero(np.all(indicators, axis=1))

    return idx_list


# Рассчет Евклидовой метрики.
def euclidean_metric(s_point, f_point):
    return m.sqrt((s_point[0] - f_point[0]) ** 2 + (s_point[1] - f_point[1]) ** 2)
```

```
# Построение множества Парето, вычисление расстояния и возврат индекса наилучшего решения.
def create_pareto_set(weights, criteria, utopia, criteria_names, names):
    assert len(criteria) == 2, "Необходимо 2 критерия."
    # Из критериев(столбцов матрицы оценок) берем координаты точек.
    points = weights[:, criteria]
    # Для каждой точки вычисляем евклидово расстояние до точки утопии -> массив расстояний.
    dist = [euclidean_metric(point, utopia) for point in points]
    # Строим график для демострации.
    x, y = np.transpose(points)
    colors = np.random.rand(len(x))
    plt.title('Множество Парето:')
    plt.scatter(x, y, c=colors)
    for i, label in enumerate(names):
        plt.annotate(label, (x[i], y[i]))
    plt.scatter(*utopia, c='red')
    plt.annotate('Точка утопии', utopia)
    plt.xticks(np.arange(0, 11))
    plt.yticks(np.arange(0, 11))
    plt.xlabel(criteria_names[0])
    plt.ylabel(criteria_names[1])
    plt.grid()
    plt.plot(x, y)
    plt.savefig('pareto.png')

    return dist
```

```
# Создание матрицы весов.
def create_weights_matrix(amount, pair_comparisons, inv_function):
    result = [[1. for _ in range(amount)] for _ in range(amount)]
    offset = 0
    for i in range(amount - 1):
        j = amount - 1 - i
        base = pair_comparisons[offset:offset + j]
        result[i][i + 1:] = base
        mirrored = [inv_function(elem) for elem in base[::-1]]
        result[j][:j] = mirrored
        offset += j

    return result



# Нормализация матрицы.
def normalized_priority_vector(matrix):
    # Вычисление среднего геометрического в каждой строке матрицы.
    # Произведение элементов строки матрицы в степени 1 / (размер матрицы).
    geometric_means = np.array([np.prod(row) ** (1. / len(matrix)) for row in
matrix])
    geometric_means_sum = np.sum(geometric_means)

    return geometric_means, geometric_means / geometric_means_sum



# Вычисление коэффициента согласованности.
def count_accordance(matrix, npv, rci):
    column_sums = np.sum(np.transpose(matrix), axis=1)
    own_value = sum(np.multiply(column_sums, npv))
```

```
cons_i = (own_value - len(npv)) / (len(npv) - 1)

return cons_i / rci
```