Отчет по Лабораторному практикуму №2

Тема: “Алгоритмы стохастической и эвристической однокритериальной оптимизации”

Студнт группы ДМП-101уцп

Ерохин Никита

Условие: Предприниматель желает приобрести автомобиль. Имеются 4 варианта покупки A,B,C и D. В качестве критериев выступают: Цена (K1), Комфортность (K2) и Экономичность (K3). Оценки парных сравнений альтернатив по каждому критерию и критериев между собой имеют вид.

Альтернативы

K1	A	B	C	D
A	1	1/5	3	1/2
B	5	1	1/3	1/7
C	1/3	3	1	3
D	2	7	1/3	1

K2	A	B	C	D
A	1	4	3	1/5
B	1/4	1	2	5
C	1/3	1/2	1	3
D	5	1/5	1/3	1

K3	A	B	C	D
A	1	3	1/7	5
B	1/3	1	6	3
C	7	1/6	1	1/5
D	1/5	1/3	5	1

Критерии

	K1	K2	K3
K1	1	3	1/2
K2	1/3	1	5
K3	2	1/5	1

рис. 1 Оценки парных сравнений альтернатив по каждому критерию и критериев

Решение:

На рис. 2 приведен исходный код программы и результат ее выполнения.

```
[1]: import numpy as np

[2]: #support functions to calculate eigenvectors and eigenweight of criteria matrix

def get_eigenvectors_array(matrix: np.array) -> np.array:
    return np.array([row.prod() ** (1 / len(row)) for row in matrix])


def get_eigenvectors_weight(eigenvectors: np.array) -> np.array:
    return np.array([item / eigenvectors.sum() for item in eigenvectors])

[6]: #input data
option_names = ['a', 'b', 'c', 'd']
k1 = np.array([
    [1, 1 / 5, 3, 1 / 2],
    [5, 1, 1 / 3, 1 / 7],
    [1 / 3, 3, 1, 3],
    [2, 7, 1 / 3, 1],
])
k2 = np.array([
    [1, 4, 3, 1 / 5],
    [1 / 4, 1, 2, 5],
    [1 / 3, 1 / 2, 1, 3],
    [5, 1 / 5, 1 / 3, 1],
])
k3 = np.array([
    [1, 3, 1 / 7, 5],
    [1 / 3, 1, 6, 3],
    [7, 1 / 6, 1, 1 / 5],
    [1 / 5, 1 / 3, 5, 1],
])
weight = np.array([
    [1, 3, 1 / 2],
    [1 / 3, 1, 5],
    [2, 1 / 5, 1],
])

[7]: #calculating criteria weight and weight weight(sounds weird)
crit_weights = [get_eigenvectors_weight(get_eigenvectors_array(x)) for x in [k1, k2, k3]]
weight_weight = get_eigenvectors_weight(get_eigenvectors_array(weight))

[9]: for i in range(len(option_names)):
    res = sum([crit_weights[j][i] * weight_weight[j] for j in range(len(crit_weights))])
    print(f"F({option_names[i]}) = {res}")

F(a) = 0.25136653682579757
F(b) = 0.26907096765413807
F(c) = 0.23496962545664285
F(d) = 0.24459287006342162
```

рис 2. Исходный код программы и результат выполнения

После подсчета массива, являющимся результатом вычисления функции полезности можно сделать вывод, что в условиях поставленной задачи самым лучшим выбором для покупателя является принятие решение о покупке автомобиля B.