

Міністерство освіти України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки

ЗВІТ

до практикуму № 4

з дисципліни

“Інформаційні технології підтримки прийняття рішень”

на тему: “Метод ELECTRE I”

Варіант 11

Виконала:

Студентка групи ІІІ-71

Каспрук Анастасія Андріївна

Київ 2021

1. Результати.

Завдання 1 (нумерація альтернатив починається з нуля).

матриця індексів узгодження C

0,000	0,643	0,679	0,696	0,839	0,482	0,661	0,768	0,589	0,589	0,768	0,732	0,589	0,625	0,500
0,375	0,000	0,804	0,750	0,750	0,589	0,393	0,679	0,589	0,679	0,821	0,821	0,679	0,589	0,411
0,429	0,339	0,000	0,429	0,518	0,321	0,304	0,536	0,429	0,464	0,643	0,571	0,518	0,429	0,250
0,429	0,339	0,750	0,000	0,750	0,571	0,393	0,446	0,536	0,732	0,536	0,554	0,589	0,482	0,375
0,339	0,339	0,518	0,589	0,000	0,446	0,393	0,393	0,357	0,500	0,536	0,286	0,500	0,518	0,161
0,518	0,625	0,714	0,643	0,643	0,000	0,375	0,589	0,607	0,696	0,643	0,732	0,589	0,589	0,375
0,661	0,607	0,821	0,607	0,786	0,661	0,000	0,821	0,714	0,893	0,696	0,696	0,643	0,679	0,482
0,411	0,357	0,554	0,554	0,786	0,411	0,446	0,000	0,339	0,571	0,607	0,232	0,554	0,464	0,143
0,411	0,554	0,589	0,464	0,643	0,500	0,464	0,661	0,000	0,786	0,643	0,804	0,589	0,482	0,375
0,411	0,321	0,804	0,679	0,643	0,500	0,250	0,429	0,571	0,000	0,536	0,536	0,589	0,357	0,321
0,268	0,179	0,393	0,607	0,750	0,357	0,304	0,625	0,357	0,500	0,000	0,446	0,696	0,286	0,179
0,268	0,321	0,429	0,571	0,714	0,357	0,304	0,768	0,554	0,589	0,571	0,000	0,661	0,500	0,143
0,411	0,321	0,607	0,589	0,786	0,446	0,446	0,482	0,411	0,679	0,500	0,357	0,000	0,446	0,446
0,554	0,500	0,607	0,643	0,786	0,643	0,625	0,857	0,518	0,643	0,732	0,589	0,571	0,000	0,286
0,768	0,786	0,839	0,768	0,839	0,661	0,750	0,857	0,625	0,679	0,875	0,911	0,696	0,768	0,000

матриця індексів неузгодження D

1,000	0,417	0,347	0,556	0,171	0,417	0,444	0,333	0,444	0,444	0,556	0,444	0,278	0,357	0,278
0,222	1,000	0,167	0,333	0,222	0,356	0,222	0,222	0,347	0,222	0,571	0,429	0,444	0,356	0,333
0,556	0,333	1,000	0,333	0,556	0,622	0,556	0,556	0,486	0,222	0,536	0,446	0,778	0,622	0,667
0,556	0,333	0,521	1,000	0,556	0,533	0,556	0,556	0,446	0,222	0,357	0,286	0,778	0,556	0,667
0,556	0,667	0,667	1,000	1,000	0,667	0,889	0,278	0,889	0,889	0,357	0,417	0,444	0,667	0,622
0,533	0,583	0,667	0,444	0,333	1,000	0,267	0,200	0,347	0,333	0,625	0,750	0,333	0,267	0,467
0,875	0,750	0,875	0,333	0,500	0,321	1,000	0,375	0,375	0,667	0,500	0,625	0,222	0,321	0,400
0,500	0,556	0,556	0,889	0,375	0,556	0,778	1,000	0,778	0,778	0,375	0,333	0,222	0,714	0,533
0,778	0,556	0,533	0,333	0,778	0,667	0,778	0,778	1,000	0,222	0,333	0,750	1,000	0,778	0,889
0,556	0,467	1,000	0,417	0,556	0,533	0,556	0,556	0,357	1,000	0,536	0,643	0,778	0,556	0,667
0,667	0,533	0,533	0,800	0,667	0,556	0,711	0,667	0,889	0,711	1,000	0,222	0,889	0,667	0,778
0,778	0,556	0,356	0,622	0,778	0,667	0,778	0,778	0,667	0,533	0,111	1,000	1,000	0,778	0,889
0,486	0,667	0,667	1,000	0,292	0,667	0,889	0,194	0,889	0,889	0,268	0,250	1,000	0,667	0,778
0,333	0,556	0,556	0,889	0,208	0,444	0,778	0,457	0,778	0,778	0,417	0,208	0,222	1,000	0,533
0,208	0,311	0,125	0,222	0,229	0,286	0,208	0,457	0,417	0,139	0,714	0,571	0,111	0,286	1,000

Значення порогів для індексів узгодження та неузгодження c, d

0,739 0,404

Відношення для порогових значень c, d:

0	0	0	0	1	0	0	1	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	1	0	0	0	0	0	0	1	0

Ядро відношення:

5 7 8 10 11 12 14

2. Постановка задачі.

Завдання

Задано множину альтернатив $\{A_1, \dots, A_{15}\}$, що оцінені за критеріями $k_1..k_{12}$. Оцінки альтернатив за усіма критеріями дано в таблиці.

Кожен критерій має ваговий коефіцієнт w_i .

Дано пару порогових значень індексів узгодження та неузгодження s, d відповідно.

Завдання 1.

Визначити підмножину найкращих альтернатив (ядро), використовуючи метод ELECTRE I (для заданих значень порогів індексів узгодження та неузгодження s, d).

Результати записати в файл **Var-X-Прізвище.txt**

Вихідний файл повинен містити:

- матрицю індексів узгодження (розмір 15×15 , вивід індексів 3 знаки після коми, елементи на головній діагоналі = 0)
- матрицю індексів неузгодження (розмір 15×15 , вивід індексів 3 знаки після коми, елементи на головній діагоналі = 1)
- значення s, d
- відношення на множині альтернатив, яке відповідає виконанню необхідної та достатньої умови для значень s, d
- ядро для відношення (розв'язок задачі для значень s, d)

Кожен із наведених елементів виводиться в форматі: текстовий рядок з назвою елемента, а потім, починаючи з наступного рядка, значення елемента).

Всі проміжні етапи виконання завдання 1 навести в звіті з достатніми поясненнями та обґрунтуваннями. Для отриманого результату (ядра) обґрунтувати виконання умов внутрішньої та зовнішньої стійкості.

Завдання 2.

Дослідницька задача. Аналіз впливу параметрів s, d на розв'язок задачі ПР.

Для методу ELECTRE I провести дослідження впливу зміни порогових значень s, d на склад і розмір ядра:

2.1. Визначення впливу зміни порогового значення d на склад та розмір ядра

Зафіксувати значення порогу $s=0.5$.

Змінюючи порогове значення d в інтервалі $(0; 0.5)$, встановити вплив на склад та розмір ядра. Результати представити у вигляді графіка.

2.2. Визначення впливу зміни порогового значення c на склад та розмір ядра

Зафіксувати значення порогу $d=0.49$.

Змінюючи порогове значення c в інтервалі $[0.5; 1]$, встановити вплив на склад та розмір ядра. Результати представити у вигляді графіка.

2.3. Визначення впливу одночасної зміни порогових значень DI та CI на склад та розмір ядра

Дослідити вплив одночасної зміни порогів значень c , d на склад та розмір ядра, починаючи від пари значень c_{\max} і d_{\min} (яка відповідає максимальному складу ядра). Виконуючи одночасну зміну порогів (збільшуючи поріг d і зменшуючи поріг c , в межах інтервалів, вказаних в п.2.1 і 2.2), встановити значення обох параметрів, при яких здійснюється зміна у складі ядра. Результати представити у вигляді графіка.

2.4. Висновки проведеного аналізу – впливу c , впливу d і впливу одночасної зміни c і d . Обґрунтувати вибір розв'язку задачі на основі проведеного дослідження.

Завдання для варіанту 11:

Таблиця оцінок альтернатив $A1-A15$ за критеріями $k1-k12$

3	8	9	7	7	10	7	4	5	4	2	6
3	6	7	4	6	9	9	10	7	3	3	4
5	3	3	3	10	10	1	9	7	6	1	1
2	3	4	9	7	7	2	4	10	10	3	2
1	8	1	10	4	7	6	1	1	6	3	2
8	7	4	6	6	2	1	10	6	10	3	8
10	8	6	5	5	3	1	9	9	1	5	6
6	8	3	1	1	4	10	5	2	3	5	3
5	1	8	8	5	2	5	6	9	8	8	4
5	3	4	3	10	2	4	6	9	7	4	2
4	2	5	3	7	7	6	5	1	5	7	3
4	1	4	8	2	8	2	7	3	7	6	4
4	10	2	3	1	5	3	9	1	10	4	2
4	8	6	5	2	6	2	8	2	6	3	8
4	9	7	9	7	9	2	9	8	10	2	6

Вагові коефіцієнти критеріїв $k1-k12$:

1 10 5 2 2 6 2 5 8 2 5 8

Значення порогів для індексів узгодження та неузгодження c , d
0.739 0.404

3. Розв'язок.

Завдання 1.

Першим кроком при вирішенні завдання є обчислення матриць індексів узгодження С та неузгодження D.

Для зручності у даній лабораторній роботі матриці С та D обчислювалися на базі матриці дельта векторів.

Матриця С

Для кожної з пар альтернатив сумувалися значення ваг тих критеріїв, що переважали або були рівними за дельта вектором, а потім ця сума ділилася на загальну суму ваг критеріїв. Тобто використовувалося наступне співвідношення.

$$C(A_i, A_k) = \frac{1}{\sum_{j=1}^m w_j} \left(\sum_{j | x_{ij} \geq x_{kj}} w_j \right)$$

Головна діагональ заповнювалася нулями.

У результаті отримали матрицю С.

матриця індексів узгодження С

0,000	0,643	0,679	0,696	0,839	0,482	0,661	0,768	0,589	0,589	0,768	0,732	0,589	0,625	0,500
0,375	0,000	0,804	0,750	0,750	0,589	0,393	0,679	0,589	0,679	0,821	0,821	0,679	0,589	0,411
0,429	0,339	0,000	0,429	0,518	0,321	0,304	0,536	0,429	0,464	0,643	0,571	0,518	0,429	0,250
0,429	0,339	0,750	0,000	0,750	0,571	0,393	0,446	0,536	0,732	0,536	0,554	0,589	0,482	0,375
0,339	0,339	0,518	0,589	0,000	0,446	0,393	0,393	0,357	0,500	0,536	0,286	0,500	0,518	0,161
0,518	0,625	0,714	0,643	0,643	0,000	0,375	0,589	0,607	0,696	0,643	0,732	0,589	0,589	0,375
0,661	0,607	0,821	0,607	0,786	0,661	0,000	0,821	0,714	0,893	0,696	0,696	0,643	0,679	0,482
0,411	0,357	0,554	0,554	0,786	0,411	0,446	0,000	0,339	0,571	0,607	0,232	0,554	0,464	0,143
0,411	0,554	0,589	0,464	0,643	0,500	0,464	0,661	0,000	0,786	0,643	0,804	0,589	0,482	0,375
0,411	0,321	0,804	0,679	0,643	0,500	0,250	0,429	0,571	0,000	0,536	0,536	0,589	0,357	0,321
0,268	0,179	0,393	0,607	0,750	0,357	0,304	0,625	0,357	0,500	0,000	0,446	0,696	0,286	0,179
0,268	0,321	0,429	0,571	0,714	0,357	0,304	0,768	0,554	0,589	0,571	0,000	0,661	0,500	0,143
0,411	0,321	0,607	0,589	0,786	0,446	0,446	0,482	0,411	0,679	0,500	0,357	0,000	0,446	0,446
0,554	0,500	0,607	0,643	0,786	0,643	0,625	0,857	0,518	0,643	0,732	0,589	0,571	0,000	0,286
0,768	0,786	0,839	0,768	0,839	0,661	0,750	0,857	0,625	0,679	0,875	0,911	0,696	0,768	0,000

Матриця D

Для обчислення значень матриці D використовувалося наступне співвідношення.

$$D(A_i, A_k) = \begin{cases} 0, & \text{якщо } \forall j \ x_{ij} \geq x_{kj}, \\ \left[\frac{\max_{j | x_{ij} < x_{kj}} \{w_j(x_{kj} - x_{ij})\}}{\max_{j | x_{ij} < x_{kj}, \forall p, s} \{w_j | x_{pj} - x_{sj} | \}} \right], & \text{якщо } \exists j \ x_{ij} < x_{kj} \end{cases}$$

Головна діагональ заповнювалася одиницями.

У результаті отримали матрицю D.

матриця індексів неузгодження D

1,000	0,417	0,347	0,556	0,171	0,417	0,444	0,333	0,444	0,444	0,556	0,444	0,278	0,357	0,278
0,222	1,000	0,167	0,333	0,222	0,356	0,222	0,222	0,347	0,222	0,571	0,429	0,444	0,356	0,333
0,556	0,333	1,000	0,333	0,556	0,622	0,556	0,556	0,486	0,222	0,536	0,446	0,778	0,622	0,667
0,556	0,333	0,521	1,000	0,556	0,533	0,556	0,556	0,446	0,222	0,357	0,286	0,778	0,556	0,667
0,556	0,667	0,667	1,000	1,000	0,667	0,889	0,278	0,889	0,889	0,357	0,417	0,444	0,667	0,622
0,533	0,583	0,667	0,444	0,333	1,000	0,267	0,200	0,347	0,333	0,625	0,750	0,333	0,267	0,467
0,875	0,750	0,875	0,333	0,500	0,321	1,000	0,375	0,375	0,667	0,500	0,625	0,222	0,321	0,400
0,500	0,556	0,556	0,889	0,375	0,556	0,778	1,000	0,778	0,778	0,375	0,333	0,222	0,714	0,533
0,778	0,556	0,533	0,333	0,778	0,667	0,778	0,778	1,000	0,222	0,333	0,750	1,000	0,778	0,889
0,556	0,467	1,000	0,417	0,556	0,533	0,556	0,556	0,357	1,000	0,536	0,643	0,778	0,556	0,667
0,667	0,533	0,533	0,800	0,667	0,556	0,711	0,667	0,889	0,711	1,000	0,222	0,889	0,667	0,778
0,778	0,556	0,356	0,622	0,778	0,667	0,778	0,778	0,667	0,533	0,111	1,000	1,000	0,778	0,889
0,486	0,667	0,667	1,000	0,292	0,667	0,889	0,194	0,889	0,889	0,268	0,250	1,000	0,667	0,778
0,333	0,556	0,556	0,889	0,208	0,444	0,778	0,457	0,778	0,778	0,417	0,208	0,222	1,000	0,533
0,208	0,311	0,125	0,222	0,229	0,286	0,208	0,457	0,417	0,139	0,714	0,571	0,111	0,286	1,000

Далі на базі матриць C та D потрібно було побудувати відношення переваги для порогових значень $c = 0.739$, $d = 0.404$. Для кожної пари альтернатив якщо значення з матриці C більше або рівне за порогове значення c , а також значення з матриці D менше або рівне за порогове значення d , то пара альтернатив включалася у відношення переваги.

У результаті отримали наступне відношення переваги для порогових значень.

Відношення для порогових значень c , d :

0	0	0	0	1	0	0	1	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	1	0	0	0	0	0	0	1	0

Останнім етапом є пошук ядра (найкращих альтернатив) даного відношення. Якщо відношення є ациклічним, пошук найкращих альтернатив найзручніше здійснювати за допомогою оптимізації за Нейманом-Моршенштерном. Якщо відношення не є ациклічним, варто підібрати інше порогове значення c .

Після відбору альтернатив, які б задовольняли умови внутрішньої та зовнішньої стійкостей для оптимізації за Нейманом-Моршенштерном, отримали ядро (нумерація альтернатив починається з нуля).

Ядро відношення:

5 7 8 10 11 12 14

Завдання 2.

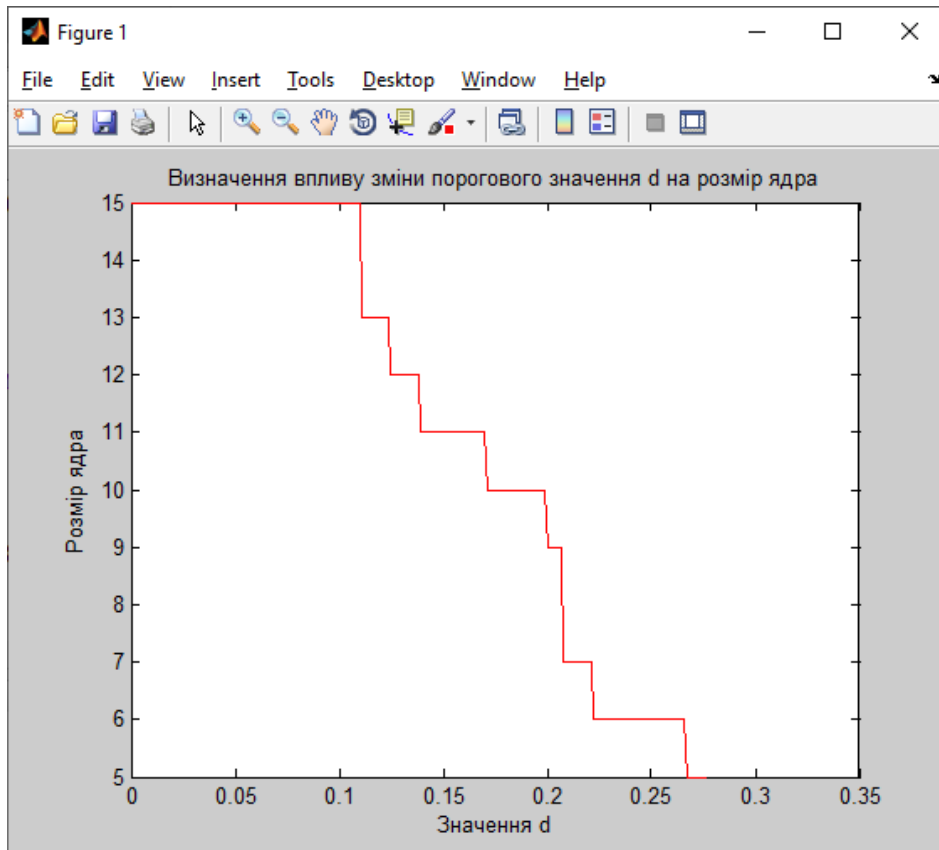
2.1. Зафіксуємо значення порогу $c = 0.5$.

Змінюючи порогове значення d в інтервалі $(0; 0.5)$, встановимо вплив на склад та розмір ядра. Значення d змінюватимемо з кроком 0.001 і виводитимемо у консоль значення, для яких склад або розмір ядра змінився.

```
Microsoft Visual Studio Debug Console
Для значення d = 0.001 розмір ядра складає 15
Поточне наповнення ядра: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
Для значення d = 0.111 розмір ядра змінився з 15 на 13
Попереднє наповнення ядра: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
Поточне наповнення ядра: 0 1 2 3 4 5 6 7 8 9 11 13 14
3 ядра були виключені значення: 10 12
Для значення d = 0.125 розмір ядра змінився з 13 на 12
Попереднє наповнення ядра: 0 1 2 3 4 5 6 7 8 9 11 13 14
Поточне наповнення ядра: 0 1 3 4 5 6 7 8 9 11 13 14
3 ядра були виключені значення: 2
Для значення d = 0.139 розмір ядра змінився з 12 на 11
Попереднє наповнення ядра: 0 1 3 4 5 6 7 8 9 11 13 14
Поточне наповнення ядра: 0 1 3 4 5 6 7 8 11 13 14
3 ядра були виключені значення: 9
Для значення d = 0.171 розмір ядра змінився з 11 на 10
Попереднє наповнення ядра: 0 1 3 4 5 6 7 8 11 13 14
Поточне наповнення ядра: 0 1 3 5 6 7 8 11 13 14
3 ядра були виключені значення: 4
Для значення d = 0.200 розмір ядра змінився з 10 на 9
Попереднє наповнення ядра: 0 1 3 5 6 7 8 11 13 14
Поточне наповнення ядра: 0 1 3 5 6 8 11 13 14
3 ядра були виключені значення: 7
Для значення d = 0.208 розмір ядра змінився з 9 на 7
Попереднє наповнення ядра: 0 1 3 5 6 8 11 13 14
Поточне наповнення ядра: 1 3 5 8 10 13 14
3 ядра були виключені значення: 0 6 11
До ядра були додані значення: 10
Для значення d = 0.222 розмір ядра змінився з 7 на 6
Попереднє наповнення ядра: 1 3 5 8 10 13 14
Поточне наповнення ядра: 1 5 8 10 13 14
3 ядра були виключені значення: 3
Для значення d = 0.267 розмір ядра змінився з 6 на 5
Попереднє наповнення ядра: 1 5 8 10 13 14
Поточне наповнення ядра: 1 5 8 11 14
3 ядра були виключені значення: 10 13
До ядра були додані значення: 11
Для значення d = 0.278 у відношенні для порогових значень 'c', 'd' був знайдений цикл, тому аналіз для поточного значення 'c' варто припинити
```

Змінюючи значення d при фіксованому значення c , ми можемо регулювати склад та розмір ядра. Як бачимо, зі збільшенням значення d розмір ядра зменшується. Переважно альтернативи з ядра лише виключаються, проте подекуди додаються. Починаючи зі значення $d = 0.278$ у відношенні переваги натрапили на цикл, тому для поточного значення c припинили подальший пошук зі збільшенням d .

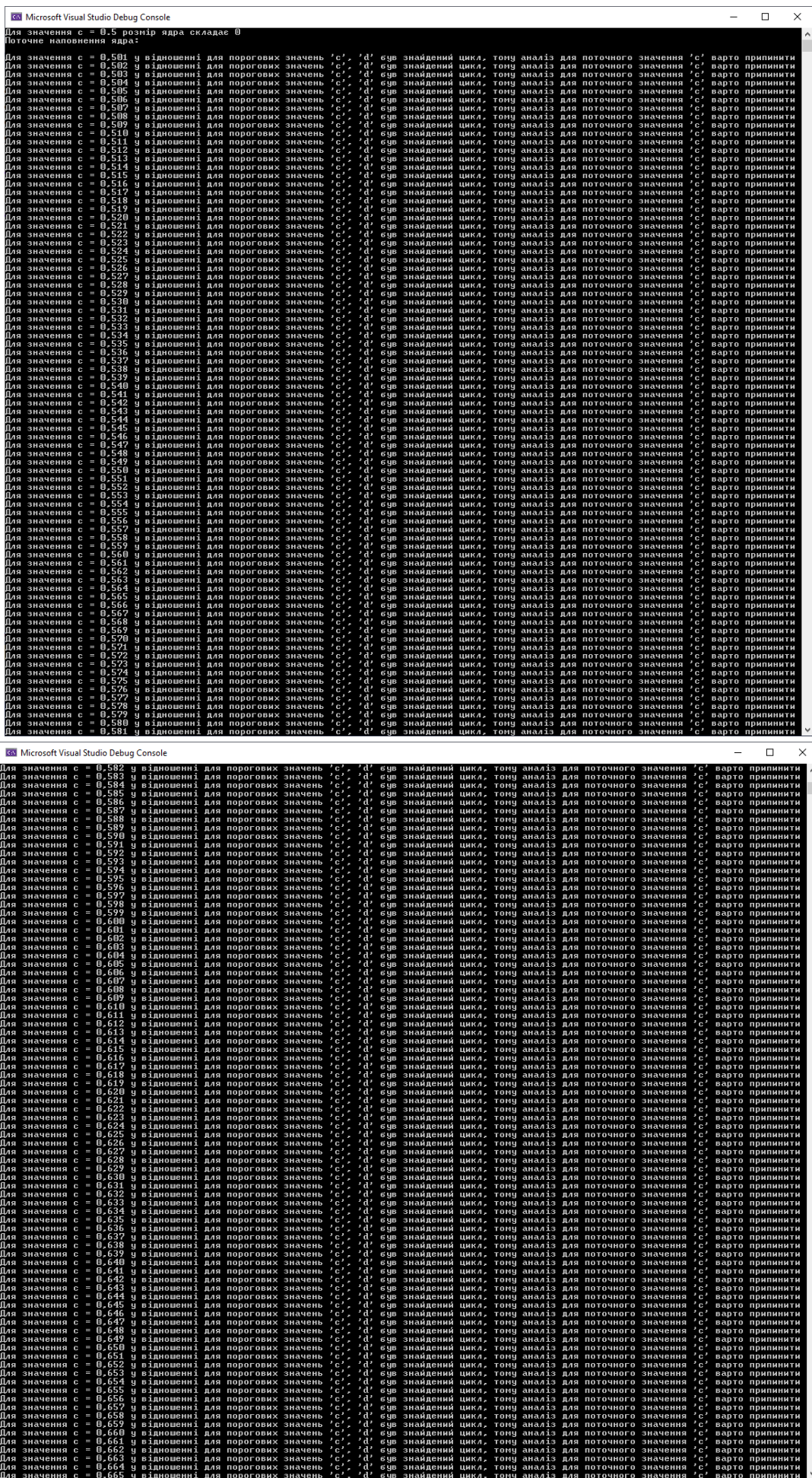
Результати у вигляді графіка.



2.2. Зафіксуємо значення порогу $d = 0.49$.

Змінюючи порогове значення c в інтервалі $[0.5; 1]$, встановимо вплив на склад та розмір ядра. Значення c змінюватимемо з кроком 0.001 і виводитимемо у консоль значення, для яких склад або розмір ядра змінився.

Для значень c в інтервалі $[0.5; 0.679]$ у відношенні переваги натрапляли на цикл і були змушені переходити до наступного значення c , що видно з виводу в консоль. Проте, починаючи зі значення $c = 0.68$ отримали ациклічне відношення переваги. Склад ядра почав поступово збільшуватися і досягнув максимуму (включення у ядро всіх альтернатив) при $c = 0.858$.



```
Microsoft Visual Studio Debug Console

Для значення c = 0.666 у відношенні для порогових значень 'с', 'd' був знайдений цикл, тому аналіз для поточного значення 'с' варто припинити
Для значення c = 0.667 у відношенні для порогових значень 'с', 'd' був знайдений цикл, тому аналіз для поточного значення 'с' варто припинити
Для значення c = 0.668 у відношенні для порогових значень 'с', 'd' був знайдений цикл, тому аналіз для поточного значення 'с' варто припинити
Для значення c = 0.669 у відношенні для порогових значень 'с', 'd' був знайдений цикл, тому аналіз для поточного значення 'с' варто припинити
Для значення c = 0.670 у відношенні для порогових значень 'с', 'd' був знайдений цикл, тому аналіз для поточного значення 'с' варто припинити
Для значення c = 0.671 у відношенні для порогових значень 'с', 'd' був знайдений цикл, тому аналіз для поточного значення 'с' варто припинити
Для значення c = 0.672 у відношенні для порогових значень 'с', 'd' був знайдений цикл, тому аналіз для поточного значення 'с' варто припинити
Для значення c = 0.673 у відношенні для порогових значень 'с', 'd' був знайдений цикл, тому аналіз для поточного значення 'с' варто припинити
Для значення c = 0.674 у відношенні для порогових значень 'с', 'd' був знайдений цикл, тому аналіз для поточного значення 'с' варто припинити
Для значення c = 0.675 у відношенні для порогових значень 'с', 'd' був знайдений цикл, тому аналіз для поточного значення 'с' варто припинити
Для значення c = 0.676 у відношенні для порогових значень 'с', 'd' був знайдений цикл, тому аналіз для поточного значення 'с' варто припинити
Для значення c = 0.677 у відношенні для порогових значень 'с', 'd' був знайдений цикл, тому аналіз для поточного значення 'с' варто припинити
Для значення c = 0.678 у відношенні для порогових значень 'с', 'd' був знайдений цикл, тому аналіз для поточного значення 'с' варто припинити
Для значення c = 0.679 у відношенні для порогових значень 'с', 'd' був знайдений цикл, тому аналіз для поточного значення 'с' варто припинити
Для значення c = 0.680 розмір ядра знівився з 0 на 5
Попереднє наповнення ядра:
Поточне наповнення ядра: 5 8 10 11 14
До ядра були додані значення: 5 8 10 11 14

Для значення c = 0.697 розмір ядра знівився з 5 на 6
Попереднє наповнення ядра: 5 8 10 11 14
Поточне наповнення ядра: 5 8 10 11 12 14
До ядра були додані значення: 12

Для значення c = 0.751 розмір ядра знівився з 6 на 7
Попереднє наповнення ядра: 5 8 10 11 12 14
Поточне наповнення ядра: 5 6 8 10 11 12 14
До ядра були додані значення: 6

Для значення c = 0.769 розмір ядра знівився з 7 на 10
Попереднє наповнення ядра: 5 6 8 10 11 12 14
Поточне наповнення ядра: 0 3 5 6 8 10 11 12 13 14
До ядра були додані значення: 0 3 13

Для значення c = 0.787 розмір ядра знівився з 10 на 11
Попереднє наповнення ядра: 0 3 5 6 8 10 11 12 13 14
Поточне наповнення ядра: 0 1 3 5 6 8 9 10 12 13 14
3 ядра були виключені значення: 11
До ядра були додані значення: 1 9

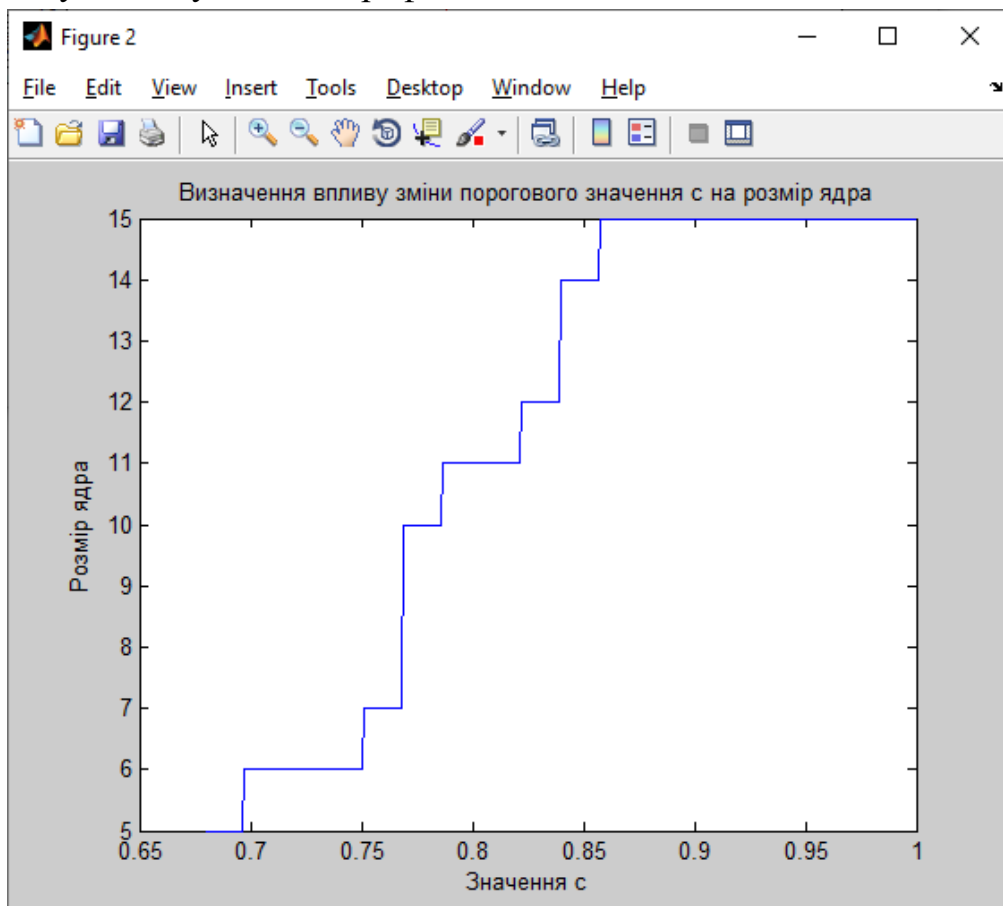
Для значення c = 0.822 розмір ядра знівився з 11 на 12
Попереднє наповнення ядра: 0 1 3 5 6 8 9 10 12 13 14
Поточне наповнення ядра: 0 1 3 5 6 8 9 10 11 12 13 14
До ядра були додані значення: 11

Для значення c = 0.840 розмір ядра знівився з 12 на 14
Попереднє наповнення ядра: 0 1 3 5 6 8 9 10 11 12 13 14
Поточне наповнення ядра: 0 1 2 3 4 5 6 8 9 10 11 12 13 14
До ядра були додані значення: 2 4

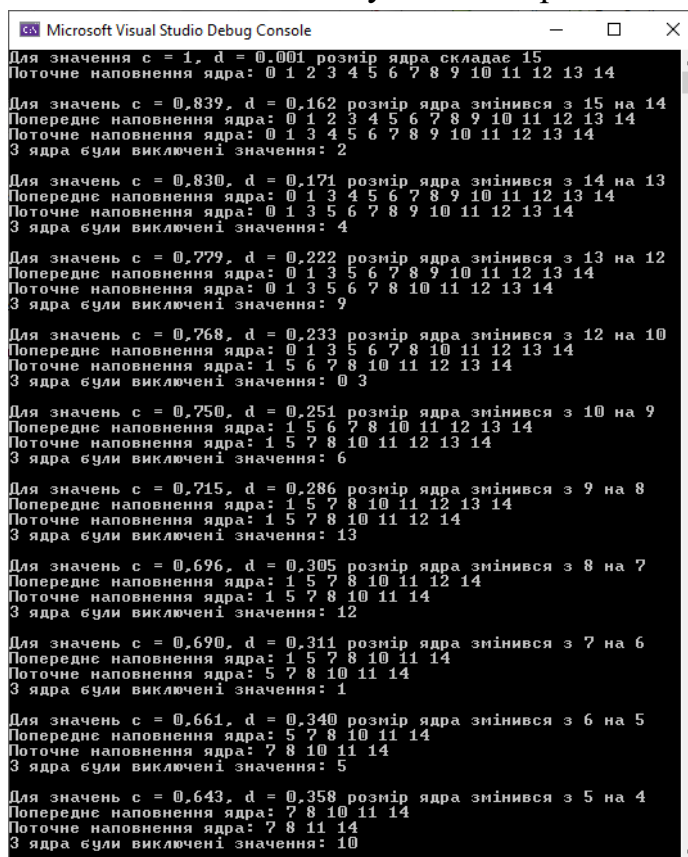
Для значення c = 0.858 розмір ядра знівився з 14 на 15
Попереднє наповнення ядра: 0 1 2 3 4 5 6 8 9 10 11 12 13 14
Поточне наповнення ядра: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
До ядра були додані значення: 7

D:\XPI\decisions\decisions\Lab4\bin\Debug\netcoreapp3.1\Lab4.exe (process 21060) exited with code 0.
```

Результати у вигляді графіка.



2.3. Дослідимо вплив одночасної зміни порогів значень c , d на склад та розмір ядра, починаючи від пари значень $c_{\max} = 1$ і $d_{\min} = 0.001$ (що відповідають максимальному складу ядра). Виконуючи одночасну зміну порогів (збільшуючи поріг d і зменшуючи поріг c з кроком 0.001, в межах інтервалів $(0; 0.5)$ і $[0.5; 1]$ відповідно), встановимо значення обох параметрів, при яких здійснюється зміна у складі ядра.



```

Microsoft Visual Studio Debug Console
Для значень c = 1, d = 0.001 розмір ядра складає 15
Поточне наповнення ядра: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Для значень c = 0.839, d = 0.162 розмір ядра змінився з 15 на 14
Попереднє наповнення ядра: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
Поточне наповнення ядра: 0 1 3 4 5 6 7 8 9 10 11 12 13 14
3 ядра були виключені значення: 2

Для значень c = 0.830, d = 0.171 розмір ядра змінився з 14 на 13
Попереднє наповнення ядра: 0 1 3 4 5 6 7 8 9 10 11 12 13 14
Поточне наповнення ядра: 0 1 3 5 6 7 8 9 10 11 12 13 14
3 ядра були виключені значення: 4

Для значень c = 0.779, d = 0.222 розмір ядра змінився з 13 на 12
Попереднє наповнення ядра: 0 1 3 5 6 7 8 9 10 11 12 13 14
Поточне наповнення ядра: 0 1 3 5 6 7 8 10 11 12 13 14
3 ядра були виключені значення: 9

Для значень c = 0.768, d = 0.233 розмір ядра змінився з 12 на 10
Попереднє наповнення ядра: 0 1 3 5 6 7 8 10 11 12 13 14
Поточне наповнення ядра: 1 5 6 7 8 10 11 12 13 14
3 ядра були виключені значення: 0 3

Для значень c = 0.750, d = 0.251 розмір ядра змінився з 10 на 9
Попереднє наповнення ядра: 1 5 6 7 8 10 11 12 13 14
Поточне наповнення ядра: 1 5 7 8 10 11 12 13 14
3 ядра були виключені значення: 6

Для значень c = 0.715, d = 0.286 розмір ядра змінився з 9 на 8
Попереднє наповнення ядра: 1 5 7 8 10 11 12 13 14
Поточне наповнення ядра: 1 5 7 8 10 11 12 14
3 ядра були виключені значення: 13

Для значень c = 0.696, d = 0.305 розмір ядра змінився з 8 на 7
Попереднє наповнення ядра: 1 5 7 8 10 11 12 14
Поточне наповнення ядра: 1 5 7 8 10 11 14
3 ядра були виключені значення: 12

Для значень c = 0.690, d = 0.311 розмір ядра змінився з 7 на 6
Попереднє наповнення ядра: 1 5 7 8 10 11 14
Поточне наповнення ядра: 5 7 8 10 11 14
3 ядра були виключені значення: 1

Для значень c = 0.661, d = 0.340 розмір ядра змінився з 6 на 5
Попереднє наповнення ядра: 5 7 8 10 11 14
Поточне наповнення ядра: 7 8 10 11 14
3 ядра були виключені значення: 5

Для значень c = 0.643, d = 0.358 розмір ядра змінився з 5 на 4
Попереднє наповнення ядра: 7 8 10 11 14
Поточне наповнення ядра: 7 8 11 14
3 ядра були виключені значення: 10

```

З виводу в консоль бачимо, що в інтервалах c від 1 до 0.643 та d від 0.001 до 0.358 ядро поступово зменшується.

Починаючи зі значень $c = 0.584$, $d = 0.417$ у відношенні переваги знаходимо цикл, що не дозволяє пошук ядра для наступних пар значень.

4. Лістинг програми.

Посилання на github-репозиторій з кодом:

<https://github.com/KasprukNastia/decisions/tree/master/Lab4>

Клас **WeightedCriteriaRelation**

```
/// <summary>
/// Клас, що описує зважене критеріальне відношення
/// </summary>
public class WeightedCriteriaRelation
{
    /// <summary>
    /// Значення критеріїв для альтернатив
    /// </summary>
    public int[][] Evaluations { get; }

    /// <summary>
    /// Ваги критеріїв
    /// </summary>
    public IReadOnlyList<double> Weights { get; }

    /// <summary>
    /// К-сть критеріїв
    /// </summary>
    public int CriteriasCount { get; }

    /// <summary>
    /// К-сть альтернатив
    /// </summary>
    public int AlternativesCount { get; }

    /// <summary>
    /// Порогове значення індекса узгодження
    /// </summary>
    public double C { get; }

    /// <summary>
    /// Порогове значення індекса неузгодження
    /// </summary>
    public double D { get; }

    /// <summary>
    /// Матриця дельта векторів
    /// </summary>
    private List<int>[][] _deltaVectors;
    /// <summary>
    /// Матриця дельта векторів
    /// </summary>
    public List<int>[][] DeltaVectors
    {
        get
        {
            if (_deltaVectors != null)
                return _deltaVectors;

            _deltaVectors = new List<int>[AlternativesCount][];
            for (int i = 0; i < AlternativesCount; i++)
                _deltaVectors[i] = new List<int>[AlternativesCount];

            for (int i = 0; i < AlternativesCount; i++)
            {
                _deltaVectors[i][i] = Enumerable.Repeat(0, AlternativesCount).ToList();
            }
        }
    }
}
```

```

        for (int j = i + 1; j < AlternativesCount; j++)
        {
            _deltaVectors[i][j] = Evaluations[i].Select((elem, index) => elem - Evaluations[j][index]).ToList();
            _deltaVectors[j][i] = _deltaVectors[i][j].Select(elem => elem * -1).ToList();
        }
    }

    return _deltaVectors;
}
}

/// <summary>
/// Матриця індексів узгодження
/// </summary>
private double[][] _cRelation;
/// <summary>
/// Матриця індексів узгодження
/// </summary>
public double[][] CRelation
{
    get
    {
        if (_cRelation != null)
            return _cRelation;

        _cRelation = new double[AlternativesCount][];
        for (int i = 0; i < AlternativesCount; i++)
            _cRelation[i] = new double[AlternativesCount];

        double weidhtsSum = Weights.Sum();
        for (int i = 0; i < AlternativesCount; i++)
        {
            _cRelation[i][i] = 0;

            for (int j = i + 1; j < AlternativesCount; j++)
            {
                _cRelation[i][j] =
                    Math.Round(DeltaVectors[i][j].Select((elem, index) => elem >= 0 ? Weights[index] : 0).Sum() / weidhtsSum, 3,
MidpointRounding.AwayFromZero);
                _cRelation[j][i] =
                    Math.Round(DeltaVectors[j][i].Select((elem, index) => elem >= 0 ? Weights[index] : 0).Sum() / weidhtsSum, 3,
MidpointRounding.AwayFromZero);
            }
        }

        return _cRelation;
    }
}

/// <summary>
/// Матриця індексів неузгодження
/// </summary>
private double[][] _dRelation;
/// <summary>
/// Матриця індексів неузгодження
/// </summary>
public double[][] DRelation
{
    get
    {
        if (_dRelation != null)
            return _dRelation;

        _dRelation = new double[AlternativesCount][];
        for (int i = 0; i < AlternativesCount; i++)

```



```

        _dRelation[i] = new double[AlternativesCount];

for (int i = 0; i < AlternativesCount; i++)
{
    _dRelation[i][i] = 1;

    for (int j = i + 1; j < AlternativesCount; j++)
    {
        _dRelation[i][j] = GetDValue(DeltaVectors[i][j]);
        _dRelation[j][i] = GetDValue(DeltaVectors[j][i]);
    }
}

return _dRelation;
}
}

/// <summary>
/// Відношення для порогових значень C, D
/// </summary>
private Relation _bestRelation;
/// <summary>
/// Відношення для порогових значень C, D
/// </summary>
public Relation BestRelation
{
    get
    {
        if (_bestRelation != null)
            return _bestRelation;

        int[][] bestRelation = new int[AlternativesCount][];
        for (int i = 0; i < AlternativesCount; i++)
            bestRelation[i] = new int[AlternativesCount];

        for (int i = 0; i < AlternativesCount; i++)
        {
            for (int j = i; j < AlternativesCount; j++)
            {
                bestRelation[i][j] = CRelation[i][j] >= C && DRelation[i][j] <= D ? 1 : 0;
                bestRelation[j][i] = CRelation[j][i] >= C && DRelation[j][i] <= D ? 1 : 0;
            }
        }

        _bestRelation = new Relation(bestRelation);

        return _bestRelation;
    }
}

/// <summary>
/// Ядро відношення
/// </summary>
private HashSet<int> _core;
/// <summary>
/// Ядро відношення
/// </summary>
public IReadOnlyCollection<int> Core
{
    get
    {
        if (_core != null)
            return _core;

        var bfsCycleFinder = new BfsCycleFinder();
        if (bfsCycleFinder.HasCycle(BestRelation))

```

```

        _core = new HashSet<int>();
    else
    {
        var optimizator = new NMOptimization();
        _core = optimizator.GetBestAlternatives(BestRelation).OrderBy(e => e).ToHashSet();

        if (!optimizator.IsBestAlternativesInternallyStable(_core, BestRelation) ||
            !optimizator.IsBestAlternativesExternallyStable(_core, BestRelation))
            throw new InvalidOperationException("Error in core calculation");
    }

    return _core;
}
}

public WeightedCriteriaRelation(int[][] evaluations,
    List<double> weights,
    double c,
    double d)
{
    Evaluations = evaluations ?? throw new ArgumentNullException(nameof(evaluations));

    AlternativesCount = evaluations.Length;
    if (AlternativesCount > 0)
        CriteriasCount = evaluations[0].Length;
    for (int i = 1; i < AlternativesCount; i++)
    {
        if (evaluations[i].Length != CriteriasCount)
            throw new ArgumentException($"Evaluation must be provided only for {CriteriasCount} criterias");
    }

    Weights = weights ?? throw new ArgumentNullException(nameof(weights));
    if (Weights.Count != CriteriasCount)
        throw new ArgumentException("The number of weights does not meet the number of criterias");

    if (c < 0 || c > 1)
        throw new ArgumentException("Value of 'c' must be between 0 and 1");
    C = c;

    if (d < 0 || d > 1)
        throw new ArgumentException("Value of 'd' must be between 0 and 1");
    D = d;
}

/// <summary>
/// Отримання індексу неузгодження для пари альтернатив на базі дельта вектору
/// </summary>
private double GetDValue(List<int> deltaVector)
{
    var indices = new List<int>();
    double nominator = deltaVector.Select((elem, index) =>
    {
        if (elem < 0)
        {
            indices.Add(index);
            return Weights[index] * -1 * elem;
        }
    }).Max();
    double denominator = indices.Count > 0 ?
        indices.Select(index => Weights[index] * GetDeltaForCriteria(index)).Max() :
        1;

    return Math.Round(nominator / denominator, 3, MidpointRounding.AwayFromZero);
}

```

```

/// <summary>
/// Отримання різниці між найбільшим та найменшим значенням для критерію
/// </summary>
private int GetDeltaForCriteria(int criteria)
{
    int[] criteriaValues = new int[AlternativesCount];
    for (int i = 0; i < AlternativesCount; i++)
    {
        criteriaValues[i] = Evaluations[i][criteria];
    }

    return criteriaValues.Max() - criteriaValues.Min();
}
}

```

Клас **BfsCycleFinder**

```

public class BfsCycleFinder
{
    // Пошук циклу у графі за допомогою алгоритму BFS
    public bool HasCycle(Relation relation)
    {
        // visited - множина для вже пройдених вершин, щоб не повертатися у них
        HashSet<int> visited = new HashSet<int>();
        // queue - черга для BFS
        Queue<int> queue;
        // currentVertex - номер вершини, витягнутої з черги BFS
        int currentVertex;
        // Проходимося по всім вершинам
        for(int vertex = 0; vertex < relation.Dimension; vertex++)
        {
            queue = new Queue<int>();

            // Додаємо всі сусідні для поточної (vertex) ще не пройдені вершини у чергу
            for (int index = 0; index < relation.Dimension; index++)
            {
                if (relation.Connections[vertex][index] == 1 && !visited.Contains(index))
                    queue.Enqueue(index);
            }

            // Поки черга не спорожніє
            while (queue.Count > 0)
            {
                currentVertex = queue.Dequeue();

                // Якщо у черзі наткнулися на вершину, що співпадає з поточною, маємо цикл
                if (currentVertex == vertex)
                    return true;

                // Додаємо всі сусідні для витягнутої (currentVertex) ще не пройдені вершини у чергу
                for (int index = 0; index < relation.Dimension; index++)
                {
                    if (relation.Connections[currentVertex][index] == 1 && !visited.Contains(index))
                        queue.Enqueue(index);
                }
            }

            // Додаємо поточну вершину у множену пройдених
            visited.Add(vertex);
        }

        return false;
    }
}

```

Клас **NMOptimization**

```
/// <summary>
/// Клас, що реалізує алгоритм Неймана-Моргенштерна
/// </summary>
public class NMOptimization
{
    /// <summary>
    /// Пошук найкращих альтернатив за алгоритмом Неймана-Моргенштерна
    /// </summary>
    public HashSet<int> GetBestAlternatives(Relation relation)
    {
        HashSet<int> allVertices = Enumerable.Range(0, relation.Dimension).ToHashSet();
        List<HashSet<int>> sSets = new List<HashSet<int>>();

        HashSet<int> prev = new HashSet<int>();
        HashSet<int> next;
        // Побудова множин Sn
        while (allVertices.Count > 0)
        {
            next = new HashSet<int>(prev);

            foreach (int vertex in allVertices)
            {
                if (relation.GetUpperSection(vertex).Except(prev).Count() == 0)
                    next.Add(vertex);
            }

            sSets.Add(next);
            prev = next;

            allVertices = allVertices.Except(next).ToHashSet();
        }

        HashSet<int> bestAlternatives = new HashSet<int>(sSets[0]);
        if (sSets.Count == 1)
            return bestAlternatives;
        // Побудова множин Qn
        for (int i = 1; i < sSets.Count; i++)
        {
            next = sSets[i];
            foreach (int vertex in next)
            {
                if (relation.GetUpperSection(vertex).Intersect(bestAlternatives).Count() == 0)
                    bestAlternatives.Add(vertex);
            }
        }

        return bestAlternatives;
    }

    /// <summary>
    /// Перевірка найкращих альтернатив на внутрішню стійкість
    /// </summary>
    public bool IsBestAlternativesInternallyStable(HashSet<int> bestAlternatives, Relation relation)
    {
        // Якщо верхній переріз якої-небудь з найкращих альтернатив містить
        // у собі іншу найкращу альтернативу, то множина розв'язків не є внутрішньо стійкою
        foreach (int alternative in bestAlternatives)
        {
            if (relation.GetUpperSection(alternative).Intersect(bestAlternatives).Count() > 0)
                return false;
        }

        return true;
    }
}
```

```

}

/// <summary>
/// Перевірка найкращих альтернатив на зовнішню стійкість
/// </summary>
public bool IsBestAlternativesExternallyStable(HashSet<int> bestAlternatives, Relation relation)
{
    HashSet<int> notBestAlternatives =
        Enumerable.Range(0, relation.Dimension)
            .Except(bestAlternatives)
            .ToHashSet();

    // Якщо верхній переріз якої-небудь з альтернатив, що не входять до найкращих, не містить
    // у собі жодної з найкращих альтернатив, то множина розв'язків не є зовнішньо стійкою
    foreach (int alternative in notBestAlternatives)
    {
        if (relation.GetUpperSection(alternative).Intersect(bestAlternatives).Count() == 0)
            return false;
    }

    return true;
}
}

```

Клас **Relation**

```

/// <summary>
/// Клас, що описує відношення
/// </summary>
public class Relation
{
    /// <summary>
    /// Зв'язки відношення
    /// </summary>
    public int[][] Connections { get; }

    /// <summary>
    /// Розмірність відношення
    /// </summary>
    public int Dimension { get; }

    private char[][] _characteristic;

    /// <summary>
    /// Характеристика відношення у множинах 'I', 'P', 'N'
    /// </summary>
    public char[][] Characteristic
    {
        get
        {
            if (_characteristic != null)
                return _characteristic;

            _characteristic = new char[Dimension][];
            for (int i = 0; i < Dimension; i++)
                _characteristic[i] = new char[Dimension];

            for (int i = 0; i < Dimension; i++)
            {
                for (int j = i; j < Dimension; j++)
                {
                    if (Connections[i][j] == 1 && Connections[j][i] == 1)
                        _characteristic[i][j] = _characteristic[j][i] = 'I';
                    else if (Connections[i][j] == 0 && Connections[j][i] == 0)

```

```

        _characteristic[i][j] = _characteristic[j][i] = 'N';
    else if (Connections[i][j] == 1 && Connections[j][i] == 0)
    {
        _characteristic[i][j] = 'P';
        _characteristic[j][i] = 'O';
    }
    else if (Connections[i][j] == 0 && Connections[j][i] == 1)
    {
        _characteristic[j][i] = 'P';
        _characteristic[i][j] = 'O';
    }
    }
}

return _characteristic;
}
}

public Relation(int[][] connections)
{
    Connections = connections ?? throw new ArgumentNullException(nameof(connections));

    Dimension = connections.Length;

    for(int i = 0; i < Dimension; i++)
    {
        if (connections[i].Length != Dimension)
            throw new ArgumentException($"{nameof(connections)} must be represented as a square matrix");

        for (int j = 0; j < Dimension; j++)
        {
            if (connections[i][j] != 0 && connections[i][j] != 1)
                throw new ArgumentException($"{nameof(connections)} must be represented only as 0 or 1 digits");
        }
    }
}

/// <summary>
/// Отримання верхнього перерізу для вершини vertex
/// </summary>
public HashSet<int> GetUpperSection(int vertex)
{
    if (vertex < 0 || vertex >= Dimension)
        throw new ArgumentException($"The vertex {vertex} does not belong to the relation");

    HashSet<int> upperSection = new HashSet<int>();
    for (int i = 0; i < Dimension; i++)
    {
        if (Connections[i][vertex] == 1)
            upperSection.Add(i);
    }

    return upperSection;
}

/// <summary>
/// Отримання нижнього перерізу для вершини vertex
/// </summary>
public HashSet<int> GetLowerSection(int vertex)
{
    if (vertex < 0 || vertex >= Dimension)
        throw new ArgumentException($"The vertex {vertex} does not belong to the relation");

    HashSet<int> lowerSection = new HashSet<int>();
    for (int i = 0; i < Dimension; i++)
    {

```



```

        if (Connections[vertex][i] == 1)
            lowerSection.Add(i);
    }

    return lowerSection;
}

/// <summary>
/// Приведення відношення до рядка
/// </summary>
public override string ToString() =>
    string.Join(Environment.NewLine, Connections.Select(arr => string.Join(' ', arr)));

/// <summary>
/// Приведення характеристики відношення до рядка
/// </summary>
public string CharacteristicToString() =>
    string.Join(Environment.NewLine, Characteristic.Select(arr => string.Join(' ', arr)));
}

```

Клас **Program**

```

class Program
{
    static void Main(string[] args)
    {
        WeightedCriteriaRelation weightedCriteriaRelation = ReadWeightedCriteriaRelation();
        WriteResults(weightedCriteriaRelation);
        Task21(weightedCriteriaRelation);
        Task22(weightedCriteriaRelation);
        Task23(weightedCriteriaRelation);
    }

    public static WeightedCriteriaRelation ReadWeightedCriteriaRelation()
    {
        string directoryPath = Directory.GetParent(Directory.GetCurrentDirectory()).Parent.Parent.FullName;
        string fileName = $"{directoryPath}\\var11_task.txt";
        string[] allFileLines = File.ReadAllLines(fileName);

        int alternativesCount = 15;
        int[][] relation = new int[alternativesCount][];
        for (int i = 1; i < alternativesCount + 1; i++)
        {
            relation[i - 1] = allFileLines[i].Split(' ')
                .Where(s => !string.IsNullOrEmpty(s))
                .Select(s => int.Parse(s))
                .ToArray();
        }

        List<double> weights = new List<double> { 1, 10, 5, 2, 2, 6, 2, 5, 8, 2, 5, 8 };

        return new WeightedCriteriaRelation(relation, weights, c: 0.739, d: 0.404);
    }

    public static void WriteResults(WeightedCriteriaRelation weightedCriteriaRelation)
    {
        string directoryPath = Directory.GetParent(Directory.GetCurrentDirectory()).Parent.Parent.FullName;
        string fileName = $"{directoryPath}\\Var11-КаспрукАнастасія.txt";

        File.WriteAllText(fileName, string.Empty);

        File.AppendAllLines(fileName,
            new List<string>
            {
                "матриця індексів узгодження C",
            }
        );
    }
}

```

```

        GetRelationString(weightedCriteriaRelation.CRelation),
        "матриця індексів неузгодження D",
        GetRelationString(weightedCriteriaRelation.DRelation),
        "Значення порогів для індексів узгодження та неузгодження c, d",
        $"{weightedCriteriaRelation.C} {weightedCriteriaRelation.D}",
        "Відношення для порогових значень c, d:",
        weightedCriteriaRelation.BestRelation.ToString(),
        "Ядро відношення:",
        string.Join(' ', weightedCriteriaRelation.Core)
    });
}

public static string GetRelationString(double[][] relation)
{
    return string.Join(
        Environment.NewLine,
        relation.Select(arr => string.Join(' ', arr.Select(elem => string.Format("{0:0.000}", elem)))));
}

public static void Task21(WeightedCriteriaRelation weightedCriteriaRelation)
{
    double c = 0.5;
    weightedCriteriaRelation = new WeightedCriteriaRelation(
        weightedCriteriaRelation.Evaluations,
        weightedCriteriaRelation.Weights.ToList(),
        c: c,
        d: 0);

    Console.WriteLine($"Для значення d = 0 розмір ядра складає {weightedCriteriaRelation.Core.Count}");
    Console.WriteLine($"Поточне наповнення ядра: {string.Join(' ', weightedCriteriaRelation.Core)}");
    Console.WriteLine();

    var bfsCycleFinder = new BfsCycleFinder();
    List<double> dValues = new List<double>() { 0 };
    List<int> sizes = new List<int>() { weightedCriteriaRelation.Core.Count };
    IReadOnlyCollection<int> currentCore = weightedCriteriaRelation.Core;
    IEnumerable<int> addedCoreAlternatives;
    IEnumerable<int> removedCoreAlternatives;
    for (double d = 0.001; d <= 0.5; d = Math.Round(d + 0.001, 3, MidpointRounding.AwayFromZero))
    {
        weightedCriteriaRelation = new WeightedCriteriaRelation(
            weightedCriteriaRelation.Evaluations,
            weightedCriteriaRelation.Weights.ToList(),
            c: c,
            d: d);

        if (bfsCycleFinder.HasCycle(weightedCriteriaRelation.BestRelation))
        {
            Console.WriteLine($"Для значення d = {string.Format("{0:0.000}", d)} у відношенні для порогових значень 'c', 'd' був знайдений цикл, тому аналіз для поточного значення 'c' варто припинити");
            break;
        }

        removedCoreAlternatives = currentCore.Except(weightedCriteriaRelation.Core);
        addedCoreAlternatives = weightedCriteriaRelation.Core.Except(currentCore);
        if(removedCoreAlternatives.Count() != 0 || addedCoreAlternatives.Count() != 0)
        {
            if (weightedCriteriaRelation.Core.Count != sizes.Last())
                Console.WriteLine($"Для значення d = {string.Format("{0:0.000}", d)} розмір ядра змінився з {sizes.Last()} на {weightedCriteriaRelation.Core.Count}");
            else
                Console.WriteLine($"Для значення d = {string.Format("{0:0.000}", d)} розмір ядра не змінився, але змінилося його наповнення");
            Console.WriteLine($"Попереднє наповнення ядра: {string.Join(' ', currentCore)}");
            Console.WriteLine($"Поточне наповнення ядра: {string.Join(' ', weightedCriteriaRelation.Core)}");
            if (removedCoreAlternatives.Count() != 0)

```

```

        Console.WriteLine($"З ядра були виключені значення: {string.Join(' ', removedCoreAlternatives)}");
        if (addedCoreAlternatives.Count() != 0)
        {
            Console.WriteLine($"До ядра були додані значення: {string.Join(' ', addedCoreAlternatives)}");
            Console.WriteLine();
        }

        dValues.Add(d);
        sizes.Add(weightedCriteriaRelation.Core.Count);
        currentCore = weightedCriteriaRelation.Core;
    }

    string directoryPath = Directory.GetParent(Directory.GetCurrentDirectory()).Parent.Parent.FullName;
    string fileName = $"{directoryPath}\\sizes-d.txt";
    File.WriteAllText(fileName, string.Empty);
    File.WriteAllLines(fileName,
        new List<string>
        {
            string.Join(' ', dValues.Select(elem => string.Format("{0:0.000}", elem).Replace(",", "."))),
            string.Join(' ', sizes)
        });
}

public static void Task22(WeightedCriteriaRelation weightedCriteriaRelation)
{
    double d = 0.49;
    weightedCriteriaRelation = new WeightedCriteriaRelation(
        weightedCriteriaRelation.Evaluations,
        weightedCriteriaRelation.Weights.ToList(),
        c: 0.5,
        d: d);

    Console.WriteLine($"Для значення c = 0.5 розмір ядра складає {weightedCriteriaRelation.Core.Count}");
    Console.WriteLine($"Поточне наповнення ядра: {string.Join(' ', weightedCriteriaRelation.Core)}");
    Console.WriteLine();

    var bfsCycleFinder = new BfsCycleFinder();
    List<double> cValues = new List<double>() { 0 };
    List<int> sizes = new List<int>() { weightedCriteriaRelation.Core.Count };
    IReadOnlyCollection<int> currentCore = weightedCriteriaRelation.Core;
    IEnumerable<int> addedCoreAlternatives;
    IEnumerable<int> removedCoreAlternatives;
    for (double c = 0.501; c <= 1; c = Math.Round(c + 0.001, 3, MidpointRounding.AwayFromZero))
    {
        weightedCriteriaRelation = new WeightedCriteriaRelation(
            weightedCriteriaRelation.Evaluations,
            weightedCriteriaRelation.Weights.ToList(),
            c: c,
            d: d);

        if (bfsCycleFinder.HasCycle(weightedCriteriaRelation.BestRelation))
        {
            Console.WriteLine($"Для значення c = {string.Format("{0:0.000}", c)} у відношенні для порогових значень 'c', 'd' був знайдений цикл, тому аналіз для поточного значення 'c' варто припинити");
            continue;
        }

        removedCoreAlternatives = currentCore.Except(weightedCriteriaRelation.Core);
        addedCoreAlternatives = weightedCriteriaRelation.Core.Except(currentCore);
        if (removedCoreAlternatives.Count() != 0 || addedCoreAlternatives.Count() != 0)
        {
            if (weightedCriteriaRelation.Core.Count != sizes.Last())
            {
                Console.WriteLine($"Для значення c = {string.Format("{0:0.000}", c)} розмір ядра змінився з {sizes.Last()} на {weightedCriteriaRelation.Core.Count}");
            }
            else
            {
                Console.WriteLine($"Для значення c = {string.Format("{0:0.000}", c)} розмір ядра не змінився, але змінилося його наповнення");
            }
        }
    }
}

```

```

        Console.WriteLine($"Попереднє наповнення ядра: {string.Join(' ', currentCore)}");
        Console.WriteLine($"Поточне наповнення ядра: {string.Join(' ', weightedCriteriaRelation.Core)}");
        if (removedCoreAlternatives.Count() != 0)
            Console.WriteLine($"З ядра були виключені значення: {string.Join(' ', removedCoreAlternatives)}");
        if (addedCoreAlternatives.Count() != 0)
            Console.WriteLine($"До ядра були додані значення: {string.Join(' ', addedCoreAlternatives)}");
        Console.WriteLine();
    }

    cValues.Add(c);
    sizes.Add(weightedCriteriaRelation.Core.Count);
    currentCore = weightedCriteriaRelation.Core;
}

string directoryPath = Directory.GetParent(Directory.GetCurrentDirectory()).Parent.Parent.FullName;
string fileName = $"{directoryPath}\\sizes-c.txt";
File.WriteAllText(fileName, string.Empty);
File.WriteAllLines(fileName,
    new List<string>
    {
        string.Join(' ', cValues.Select(elem => string.Format("{0:0.000}", elem).Replace(",", "."))),
        string.Join(' ', sizes)
    });
}

public static void Task23(WeightedCriteriaRelation weightedCriteriaRelation)
{
    double c = 0.5;
    weightedCriteriaRelation = new WeightedCriteriaRelation(
        weightedCriteriaRelation.Evaluations,
        weightedCriteriaRelation.Weights.ToList(),
        c: c,
        d: 0);

    Console.WriteLine($"Для значення c = 0.5, d = 0 розмір ядра складає {weightedCriteriaRelation.Core.Count}");
    Console.WriteLine($"Поточне наповнення ядра: {string.Join(' ', weightedCriteriaRelation.Core)}");
    Console.WriteLine();

    var bfsCycleFinder = new BfsCycleFinder();
    List<double> cValues = new List<double>() { 0.5 };
    List<double> dValues = new List<double>() { 0 };
    List<int> sizes = new List<int>() { weightedCriteriaRelation.Core.Count };
    IReadOnlyCollection<int> currentCore = weightedCriteriaRelation.Core;
    IEnumerable<int> addedCoreAlternatives;
    IEnumerable<int> removedCoreAlternatives;
    for (double d = 0.001; d <= 0.5; d = Math.Round(d + 0.001, 3, MidpointRounding.AwayFromZero))
    {
        c = Math.Round(c + 0.001, 3, MidpointRounding.AwayFromZero);

        weightedCriteriaRelation = new WeightedCriteriaRelation(
            weightedCriteriaRelation.Evaluations,
            weightedCriteriaRelation.Weights.ToList(),
            c: c,
            d: d);

        if (bfsCycleFinder.HasCycle(weightedCriteriaRelation.BestRelation))
        {
            Console.WriteLine($"Для значень c = {string.Format("{0:0.000}", c)}, d = {string.Format("{0:0.000}", d)} у відношенні для порогових значень 'c', 'd' був знайдений цикл, тому аналіз для поточного значення 'c' варто припинити");
            continue;
        }

        removedCoreAlternatives = currentCore.Except(weightedCriteriaRelation.Core);
        addedCoreAlternatives = weightedCriteriaRelation.Core.Except(currentCore);
        if (removedCoreAlternatives.Count() != 0 || addedCoreAlternatives.Count() != 0)
        {

```

```

        if (weightedCriteriaRelation.Core.Count != sizes.Last())
            Console.WriteLine($"Для значень c = {string.Format("{0:0.000}", c)}, d = {string.Format("{0:0.000}", d)} розмір ядра змінився з {sizes.Last()} на {weightedCriteriaRelation.Core.Count}");
        else
            Console.WriteLine($"Для значень c = {string.Format("{0:0.000}", c)}, d = {string.Format("{0:0.000}", d)} розмір ядра не змінився, але змінилося його наповнення");
        Console.WriteLine($"Попереднє наповнення ядра: {string.Join(' ', currentCore)}");
        Console.WriteLine($"Поточне наповнення ядра: {string.Join(' ', weightedCriteriaRelation.Core)}");
        if (removedCoreAlternatives.Count() != 0)
            Console.WriteLine($"З ядра були виключені значення: {string.Join(' ', removedCoreAlternatives)}");
        if (addedCoreAlternatives.Count() != 0)
            Console.WriteLine($"До ядра були додані значення: {string.Join(' ', addedCoreAlternatives)}");
        Console.WriteLine();
    }

    cValues.Add(c);
    dValues.Add(d);
    sizes.Add(weightedCriteriaRelation.Core.Count);
    currentCore = weightedCriteriaRelation.Core;
}

string directoryPath = Directory.GetParent(Directory.GetCurrentDirectory()).Parent.Parent.FullName;
string fileName = $"{directoryPath}\\sizes-cd.txt";
File.WriteAllText(fileName, string.Empty);
File.WriteAllLines(fileName,
    new List<string>
    {
        string.Join(' ', cValues.Select(elem => string.Format("{0:0.000}", elem).Replace(",", "."))),
        string.Join(' ', dValues.Select(elem => string.Format("{0:0.000}", elem).Replace(",", "."))),
        string.Join(' ', sizes)
    });
}
}

```

5. Опис класів. Перелік розроблених функцій на методів.

Клас	Властивість	Опис	Тип значення, що повертає
WeightedCriteriaRelation	Evaluations	Значення критеріїв для альтернатив	int[][]
	Weights	Ваги критеріїв	ICollection<double>
	CriteriasCount	К-сть критеріїв	int
	AlternativesCount	К-сть альтернатив	int
	C	Порогове значення індекса узгодження	double
	D	Порогове значення індекса неузгодження	double
	DeltaVectors	Матриця дельта векторів	List<int>[][]
	CRelation	Матриця індексів узгодження	double[][]
	DRelation	Матриця індексів неузгодження	double[][]
	BestRelation	Відношення для порогових значень C, D	Relation
	Core	Ядро відношення	ICollection<int>

Клас	Ф-ція/Метод	Параметри	Опис	Значення, що повертає
WeightedCriteriaRelation	GetDValue	List<int> deltaVector – дельта вектор для пари альтернатив	Отримання індексу неузгодження для пари альтернатив на базі дельта вектору	double – значення індексу неузгодження для пари альтернатив
	GetDeltaForCriteria	int criteria – номер критерію	Отримання різниці між найбільшим та найменшим значенням для критерію	int - значення різниці між найбільшим та найменшим значенням для критерію
BfsCycleFinder	HasCycle	Relation relation – відношення, для якого шукається цикл	Пошук циклу у графі за допомогою алгоритму BFS	bool – булеве значення, чи має відношення цикл
NMOptimization	GetBestAlternatives	Relation relation – відношення, для якого шукаються найкращі альтернативи	Пошук найкращих альтернатив за алгоритмом Неймана-Моргенштерна	HashSet<int> - множина найкращих альтернатив за Нейманом-Моргенштерном
	IsBestAlternativesInternallyStable	HashSet<int> bestAlternatives – множина найкращих альтернатив для	Перевірка найкращих альтернатив на внутрішню стійкість	bool – булеве значення, чи є найкращі альтернативи

		перевірки Relation relation – задане відношення		внутрішньо стійкими
	IsBestAlternativesExternallyStable	HashSet<int> bestAlternatives – множина найкращих альтернатив для перевірки Relation relation – задане відношення	Перевірка найкращих альтернатив на зовнішню стійкість	bool – булеве значення, чи є найкращі альтернативи зовнішньо стійкими
Relation	GetUpperSection	int vertex – номер вершини	Отримання верхнього перерізу для вершини	HashSet<int> - верхній переріз
	GetLowerSection	int vertex – номер вершини	Отримання нижнього перерізу для вершини	HashSet<int> - нижній переріз

6. Висновки.

У даній лабораторній роботі ми мали можливість навчитися розв'язувати задачі багатокритеріальної оптимізації застосовуючи метод ELECTRE I.

Для застосування методу ELECTRE I необхідно задати вхідні дані, першим елементом яких є матриця «критерій - альтернатива». Дана матриця містить оцінки всіх альтернатив за кожним критерієм.

Далі необхідно задати вагу кожного критерію.

Заключним елементом вхідних даних методу ELECTRE I є рівень узгодження s і рівень неузгодження d , які будуть використані для визначення наявності або відсутності переваги між альтернативами.

Згідно з методом ELECTRE I, відносини переваги між альтернативами позначаються за допомогою бінарного відношення.

Для того щоб пара альтернатив була включена до відношення переваги, необхідне виконання двох умов. Індекс узгодження для пари повинен бути більшим або дорівнювати заданому пороговому індексу узгодження s , індекс неузгодження повинен бути меншим або дорівнювати заданому пороговому індексу неузгодження d . Індеси узгодження та неузгодження для пар альтернатив визначаються за спеціальними співвідношеннями, наведеними у пункті 3.

На основі отриманого бінарного відношення переваги можна знайти ядро (рішення) для задачі застосовуючи алгоритм оптимізації за Нейманом-Моргенштерном (для ациклічних відношень). Якщо отримане у результаті розв'язку відношення не є ациклічним варто змінити значення індекса узгодження s .

Розмір та склад ядра можна регулювати, змінюючи значення порогових індексів узгодження s та неузгодження d , у чому ми мали змогу переконатися виконуючи завдання 2. Для фіксованого значення індекса узгодження s зі збільшенням значення індекса неузгодження d розмір ядра зменшується (завдання 2.1.). Для фіксованого значення індекса неузгодження d , навпаки, зі збільшенням значення індекса узгодження s розмір ядра збільшується (завдання 2.2.). При одночасній зміні пари, збільшуючи поріг d і зменшуючи поріг s з фіксованим кроком в межах інтервалів $(0; 0.5)$ і $[0.5; 1]$ відповідно, розмір ядра поступово зменшується (завдання 2.3.). З проведеного аналізу можна зробити висновок, що індеси узгодження та неузгодження для розв'язку завдання 1 (а саме $s = 0.739$, $d = 0.404$) у даному варіанті є доволі прийнятними для отримання ядра оптимального розміру і складу.

У результаті ми навчилися розв'язувати задачі багатокритеріальної оптимізації застосовуючи метод ELECTRE I та змогли знайти рішення задачі у відповідності з варіантом.